AD-A268 128

MENTATION PAGE

Form Approved
OMB No. 0704-0188

REPORT DATE	3. REPORT TYPE AND DATES COVERED
April 1993	THESIS/DISSERTATION

4. TITLE AND SUBTITLE
Numerical Solution of a Second Order Boundary Problem for Two Dimensions Using Galerkin Approximations.

5. FUNDING NUMBERS

6. AUTHOR(S)
Craig L. Arnold

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Student Attending: Univ of Central Florida

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA- 93-058

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distribution Unlimited
MICHAEL M. BRICKER, SMSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES 139
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NUMERICAL SOLUTION OF A SECOND ORDER BOUNDARY VALUE PROBLEM FOR TWO DIMENSIONS USING GALERKIN APPROXIMATION METHODS

by

CRAIG L. ARNOLD
B.S.C.E., University of Nebraska, 1989

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science in Structures and Foundations
College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1993

ABSTRACT

This report outlines the development and use of the program "LAPLACE". LAPLACE is capable of solving a second order two dimensional boundary value problem, employing graphics to assist in mesh generation and solution presentation.

Galerkin approximation methods, along with the development of a finite element mesh, permit the program to calculate nodal results over the domain of the problem. The use of these nodal solutions with additional subroutines allows for the computation of equipotential lines and lines perpendicular to the equipotential lines.

The current format of this program solves Laplace's Equation. Nodal solutions to Laplace's Equation are calculated over the domain of the problem and used as the basis for the generation of equipotential lines and their perpendiculars. Equipotential lines are interpreted as contours and their perpendiculars represent flow lines for the solution to Laplace's Equation. These lines are used in combination to develop a flow net over the domain of the problem and this flow net is graphically displayed.

This program was written with the capability of solving several types of second order two dimensional boundary value problems. The calculation of solutions to other second order two dimensional boundary value problems is accomplished by entering the appropriate functional coefficients of the differential equation into one subroutine.

ACKNOWLEDGEMENT

I am deeply indebted to Dr. William F. Carroll for his continued support. His encouragement and guidance greatly helped in the timely completion of this research report. I wish to thank Dr. Jenkins and Dr. Kunnath for their instruction on topics that led to the development of this report. I enjoyed my studies at the University of Central Florida and have a great deal of respect for the entire Civil Engineering Department.

I also would like to thank my wife, Lyneé, for her understanding of the time taken to complete this report and continuing to support me through all my endeavors.

PREFACE

The study of the two dimensional second order boundary value problem is presented in the following paper. A solution to a problem of this type varies from system to system, and requires extensive work to calculate each solution. This paper outlines the reduction of equations used in the development of a numerical approximate solution and the generation of computer code that calculates the solution. The procedure has been written into computer program code capable of calculating solutions for two dimensional second order boundary value problems, and representing the results graphically. The graphics are displayed on the computer screen but can be printed to develop a better use of the program.

Chapter one describes the problem encountered in obtaining a solution to the second order differential equation and the varied procedures used calculating a solution. An overview of the steps taken to solve the second order differential equation is given.

Chapter two explains the procedure used by the program. The ability for the analytical approach to be converted to a numerical solution and represented by computer code is also discussed.

Chapter three discusses an overview of the program LAPLACE. A general explanation of the procedures followed by the program in the calculation of a solution is given and the development of numerical solutions within the program is also discussed.

Chapter four illustrates a two dimensional boundary value problem. The chapter also outlines a procedure for entering and running this example problem using LAPLACE. The program user can follow this example to familiarize himself with the operation of the program, and compare his results to those contained in appendix B.

Chapter five explains the advantages and disadvantages of this program in calculating the solution of a two dimensional second order boundary value problem. The limitations of the program are also discussed and guidelines are suggested to obtain better results from the program. Suggestions are made on the alternate uses for the program, and the possibility of developing this program into a more useful tool as memory and speed of personal computers continues to increase.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Many physical phenomena are described by the solution of an ordinary or partial differential equation. The development of solutions to these equations aids engineers in understanding physical behavior, and results in the development of better systems to control these phenomena. Differential equations describing physical behavior are varied but include elliptic, parabolic, and hyperbolic equations.[4]*

Elliptic equations usually describe the equilibrium state of a physical system. Parabolic or hyperbolic equations generally represent time dependent processes. Equilibrium systems are studied in this paper and the equations evaluated will be elliptic.[4]

Introducing different boundary conditions and regions further complicates the calculation of solutions for these differential equations. General boundary conditions for elliptic equations fall into three categories; essential, natural, or a combination of the two. Each of the three types of conditions will also be included in the development of the program and are discussed below.

Solutions to a boundary value problem can be calculated by knowing the types of equations describing the system and boundary conditions applied to the system. Boundary conditions must be known; either they are measured from a system directly or specified

* Indicates reference number in the list of references.

in the calculation. These may then be used to solve the differential equations for an exact solution. The result is a series of calculations, using a given set of data, that finds the best equation that can produce a good approximate solution for a more complicated, but real problem.

General Second Order Two Dimensional Partial Differential Equation

The intent of the program developed, was to create a program that calculates a solution to the general second order, two dimensional partial differential equation. This equation is shown in equation 1.1.

$$-\nabla\,[k(x,y)\ \nabla u]\ +\ c(x,y)\left[\frac{d\,u}{dx}\ +\ \frac{d\,u}{dy}\right]\ +\ b(x,y)\,[u]\ =\ f(x,y) \qquad 1.1$$

- **u** is a state variable (a scalar quantity which must be solved for)
- **k**, **c**, **b**, and **f** are functional coefficients which must be known
- $\mathbf{k} \neq \mathbf{0}$ and cannot change sign

By definition:

$$\nabla u \equiv \frac{\partial u}{\partial x}\ +\ \frac{\partial u}{\partial y} \qquad 1.2$$

The program is structured to incorporate the functional coefficients k, c, b, and f in an independent subroutine for ease in changing them to represent different partial differential equations.

The solution of the partial differential equations requires that the boundary conditions be known and also the type of equation (elliptic, parabolic, or hyperbolic) that defines the solution of the problem. Boundary conditions include essential, natural, and a combination of essential and natural. An essential boundary condition can be thought of as a value of the state variable on a boundary. Then a natural boundary condition would represent the change of the state variable with respect to distance, and be proportional to the rate of change across the boundary. A combination of these at different boundaries could also be specified.

Finite Element Analysis

A significant problem in the sol ɑion of a boundary value problem is the examination of the internal conditions that allow the boundary conditions to develop. Solutions to these problems may be calculated using a finite element method, determining incremental changes in the state variable throughout the problem, and assigning values to nodes or points on the interior of the domain. [4]

Problem solutions are calculated at nodal points throughout the domain of the problem; they can also be linearly interpolated between nodal values. The interpolated points of equal value are connected as contours to graphically represent the problem. These may be called equipotential lines. Perpendiculars to these lines represent the gradient of change at a particular point. The equipotential lines and their perpendiculars are arranged in a criss-cross fashion over the domain of the problem to form a mesh or net that describes the behavior of the system.

Numerical Analysis

The majority of partial differential equations that arise in the studies of practical problems are too complex for exact analytical treatment. Approximate techniques have been developed to solve them for particular solutions. Approximate integration techniques, which can be numerical or semi-analytical are extensively employed in these solutions. Numerical solutions to differential equation problems allow computers to do extensive calculations and quickly calculate an approximate solution with good accuracy. The types of numerical integration employed vary and will be discussed in chapter two.

Program Testing and Evaluation

The current edition of the program solves Laplace's Equation. The program can be modified as stated above by changing one subroutine, thus allowing for the solution to other forms of second order linear elliptic scalar partial differential equations. Laplace's Equation is used as a benchmark to determine capabilities of the program.

The solution represented by Laplace's equation is that of fluid flow over a two dimensional domain of the independent variables, x and y. In this specific case the equipotential lines connect points of equal pressure (the state variable) and are represented as contour lines through the domain. Lines drawn perpendicular to the contours are flow lines and describe fluid flow through the domain.

CHAPTER 2

THEORETICAL BACKGROUND

This program was written to aid in the solution of a general second order linear elliptic equation. The equation below is used to calculate the solution of two dimensional boundary value problems and is repeated here from chapter one.

$$-\nabla\ [k(x,y)\nabla\ u]\ +\ c(x,y)\left[\frac{du}{dx}\ +\ \frac{du}{dy}\right]\ +\ b(x,y)\ [u]\ =\ f(x,y) \qquad 2.1$$

- **u** is a state variable (a scaler quantity which must be solved for)
- **k, c, b,** and **f** are functional coefficients which must be known
- $\mathbf{k} \neq \mathbf{0}$ and cannot change sign

By definition:

$$\nabla u = \left[\ \frac{\partial u}{\partial x}\ +\ \frac{\partial u}{\partial y}\ \right] \qquad 2.2$$

The program allows for the solution of different second order partial differential equations, of the form shown in equation 2.1, by changing the coefficients for k(x,y), c(x,y), b(x,y), and f(x,y) in a subroutine. The general second order equation (2.1) can be reduced to other well known elliptic equations such as Poisson's and Laplace's Equation. By letting the following values, k(x,y) = -1, c(x,y) = 0 and b(x,y) = 0, the

general form of equation 2.1 reduces to Poisson's Equation as shown in 2.3a or 2.3b.

$$\frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y} = f(x,y) \qquad 2.3a$$

or

$$\nabla^2 u = f(x,y) \qquad 2.3b$$

The second order elliptic equation can be further reduced to Laplace's Equation by also setting f(x,y) = 0, resulting in equation 2.4a or 2.4b .

$$\frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y} = 0 \qquad 2.4a$$

or

$$\nabla^2 u = 0 \qquad 2.4b$$

Most practical problems do not require a general solution of the second order boundary value problem describing it. More commonly a particular solution satisfying specific boundary conditions is needed. LAPLACE can solve the second order partial for three types of boundary conditions: essential, natural, and a combination of essential and natural. The boundary conditions will be developed and explained throughout this chapter. The explanation for the calculation of an approximate numerical solution to a second order partial differential equation will be divided into three phases: a model problem, general one dimensional boundary value problems, and two dimensional boundary value problems.

The majority of theory and mathematics discussed in the remainder of this chapter

are contained in references [1,4]. The development of finite element solutions has been widely studied and additional information on this subject was gathered from other references, these are contained in the list of references.

Model Problem

The application of the one dimensional case can be studied by considering the following model boundary value problem:

$$-u'' + u = x \qquad 2.5$$

Over the domain:

$$0 < x < 1$$

for the following boundary conditions:

$$u(0) = 0\,, \qquad u(1) = 0$$

Equation 2.5 will be altered by introducing a variable 'v', multiplied by both sides and also integrating the equation over the domain of x. This procedure results in equation 2.6 shown below. [1]

$$\int_0^1 (-u'' + u)\, v\, dx = \int_0^1 xv\, dx \qquad 2.6$$

The test function 'v' has to satisfy the integral; forms that do not satisfy this integral will not be considered in the set of solutions that satisfy the original equation (2.5). The group of functions that do satisfy equation 2.6 will be considered in the set

'H'. Equation 2.6 is integrated by parts and the solution to this integration is shown in equation 2.7. [1]

$$\int_0^1 -u''v\,dx = \int_0^1 u'v'\,dx - u'v\Big|_0^1 \qquad 2.7$$

Substituting the results of equation 2.7 into equation 2.6 results in the development of equation 2.8. [1]

$$\int_0^1 (-u'v' + uv)\,dx = \int_0^1 xv\,dx + u'v\Big|_0^1 \quad where \quad v \subseteq H_0^1 \qquad 2.8$$

Equation 2.8 is a variational form of equation 2.5. The variational form only requires that the first derivative of the solution be definable over the domain. The original equation required that the second derivative of the solution be definable over the domain. This makes the solution less restrictive and allows the specific solution to come from a larger set of possible solutions. The development of a variational form of the original equation will also aid in the calculation of a numerical solution to the boundary value problem.

The last term of equation 2.8 represents natural boundary conditions. If natural boundary conditions are present they will be added to the end nodes or boundary of the linear problem. This term is dropped if the first derivative of the state variable, u, is equal to zero.

Equation 2.9 is the specific case of equation 2.8 where the first derivative along the boundary is zero. [1]

$$\int_0^1 (-u'v' + uv)\, dx = \int_0^1 xv\, dx \quad \text{where} \quad v \subseteq H_0^1 \qquad 2.9$$

The remaining discussion of the model problem will not include natural boundary conditions. Natural boundary conditions will be reintroduced in the one and two dimensional discussions presented later.

Galerkin Approximation

'H' in equation 2.9 represents the set of admissible functions that satisfy the equation. Two fundamental properties of 'H' are as follows:

a. 'H' is a linear space, therefore a linear combination of functions that make up the domain 'H' also satisfy the equation.

b. 'H' is infinite dimensional. It is necessary to specify an infinity of parameters to uniquely define a test function v in the domain.

To obtain a solution, v(x) is represented as a summation of an infinite set of basis functions satisfying equation 2.9. Each of the basis functions is represented by $\phi(x)$ and the value for v(x) can be expressed by the summation shown in equation 2.10.

$$v(x) = \sum_{i=1}^{\infty} \beta_i \phi_i (x) \qquad 2.10$$

β_i in equation 2.10 represents constants and the series must converge. Thus an approximation of the solution can be represented by a finite number of terms. This approximate solution is represented by the finite series as displayed in equation 2.11.

$$v_N(x) = \sum_{i=1}^{N} \beta_i \phi_i (x) \qquad 2.11$$

The Galerkin method seeks an approximate solution to the boundary value problem, calculated by considering the solution in a sub space of 'H' rather than the entire solution over the whole set of 'H'. Equation 2.9 can be rewritten as shown in equation 2.12 and the set of solutions that satisfy the differential equation is now reduced.

$$\int_0^1 (-u_N' v_N' + u_N v_N)\, dx = \int_0^1 x v_N\, dx \quad where \quad v \subseteq H_0^N \qquad 2.12$$

The domain is divided into sections and the sum of the equations over these sections or elements represent the solution, equation 2.13 can be defined, and α_i represents the degrees of freedom in the problem.

$$u_N(x) = \sum_{i=1}^{N} \alpha_i \phi_i (x) \qquad 2.13$$

The basis functions ϕ_i are predetermined by using functions that satisfy the boundary conditions of the problem. Replacing v_n and u_n in equation 2.12 with the respective summations (2.11 and 2.13) the equation 2.14 can be determined.

$$\sum_{i=1}^{N} \beta_i \left[\sum_{j=1}^{N} \left(\int_0^1 [\phi_i'(x)\phi_j'(x) + \phi_i(x)\phi_j(x)]\, dx \right) \alpha_j - \int_0^1 x\phi_i(x)\, dx \right] = 0 \qquad 2.14$$

Equation 2.14 can be rewritten into the form of equation 2.15. [1]

$$\sum_{i=1}^{N} \beta_i \left(\sum_{j=1}^{N} K_{ij}\alpha_i - F_i \right) = 0 \qquad 2.15$$

Where:

$$K_{ij} = \int_0^1 [\phi_i'(x)\,\phi_j'(x) + \phi_i(x)\,\phi_j(x)]\, dx \qquad 2.16$$

and:

$$F_i = \int_0^1 x\,\phi_i\, dx + \phi_i\, x\Big|_0^1 \qquad 2.17$$

for i , j = 1 , 2 ..., N

Due to the arbitrarity of β_i the solution of equation 2.15 represents a series of functions that satisfy the boundary value problem resulting in equation 2.18

$$\sum_{j=1}^{N} K_{i\,j}\alpha_j = F_i \qquad 2.18$$

for i, j = 1, 2,...,N

The values of 'α_j' obtained from solution of equation 2.18 are the approximate values of the state variable **u** as shown in equation 2.9.

Through Galerkin approximation methods the model problem has been reduced to the summation of equations. Calculation of each of these summations is required for

α_i to be calculated. Equation 2.18 represents the summation of the solution over the domain. The actual calculation of α_i will require matrix algebra and finite element methods.

Finite Element Basics

The finite element method allows a systematic approach for the construction of basis functions over the domain of the problem. The domain of the problem is divided into subregions known as elements. Basis functions are constructed over the entire domain of the problem from shape functions which are defined on each element. The next section will describe the basis functions used in solving this equation. These elements are represented by nodes located at their ends. The solution to the overall (global) problem is the values of the state variable at the nodes. These values are the weights in the representation of the solution to the problem, as the weighted linear combination of the basis (or shape) functions. By careful placement of the nodes on the corners or sides of individual elements, global continuity of the solution can be ensured.

Basis Functions

The following criteria have to be satisfied in the selection of basis functions that describe the solution over the domain of the problem. [1]

1. Basis functions are generated by simple functions (shape functions) defined piecewise - element by element - over the finite element mesh.

2. Basis functions are smooth enough to be members of the class H_0^1 of test functions.
3. Basis functions are chosen in such a way that the parameters defining the approximate solution u_h are the values of $u_h(x)$ at the nodal points.

These functions are normally polynomials, since they are easier to integrate than trigonometric basis functions. Generally the polynomial basis functions provide a good approximation of the exact solution.

The simplest set of basis functions that satisfy the conditions above are a set of linear piecewise segments that have a value of one at the node being evaluated, and zero at all other nodes in the domain.

These are represented by the following equations:[1]

$$\phi_i(x) = \frac{x-x_{i-1}}{h_i} \qquad for\ x_{i-1} \le x \le x_i \qquad 2.19a$$

$$\phi_i(x) = \frac{x_{i+1}-x}{h_{i+1}} \qquad for\ x_i \le x \le x_{i+1} \qquad 2.19b$$

$$\phi_i(x) = 0 \qquad for:\ x \le x_{i-1}\ ,\ x \ge x_{i+1} \qquad 2.19c$$

Equations 2.19 are graphically represented in figure 2.1 for i = 1. Notice that the basis function has a value of one at x = 1 and zero at x = 0, 2, 3, and 4. The function is linear between nodes x = 0 and x = 2 allowing for the calculation of a first derivative.

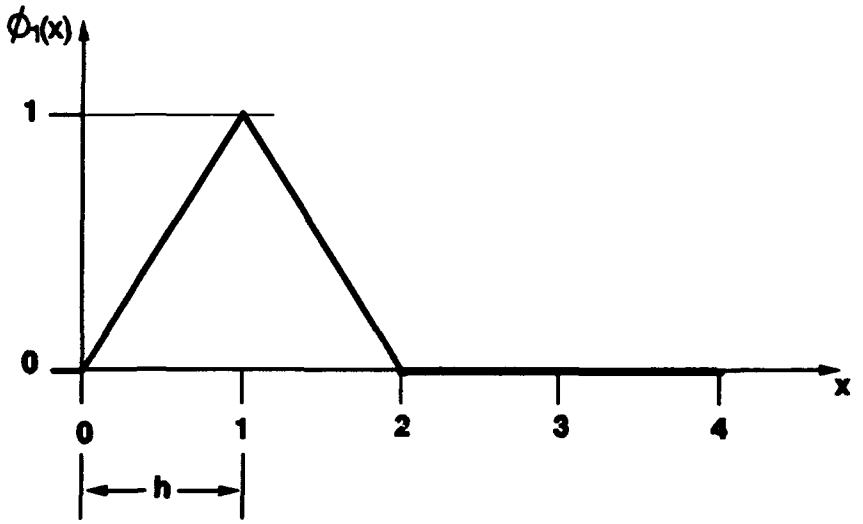


Figure 2.1
Finite Element Basis Function

Figure 2.2 may be viewed as being composed of three separate basis functions or six piece-wise shape functions each defined over the length 'h'. The conditions of the basis function are satisfied by the shape functions and their equations have a first derivative over the domain of the problem. The values for Φ and Φ' can be determined and the solution to the problem calculated by equation 2.18. A procedure for numerical integration is also used in the calculation of the solution to equation 2.18.

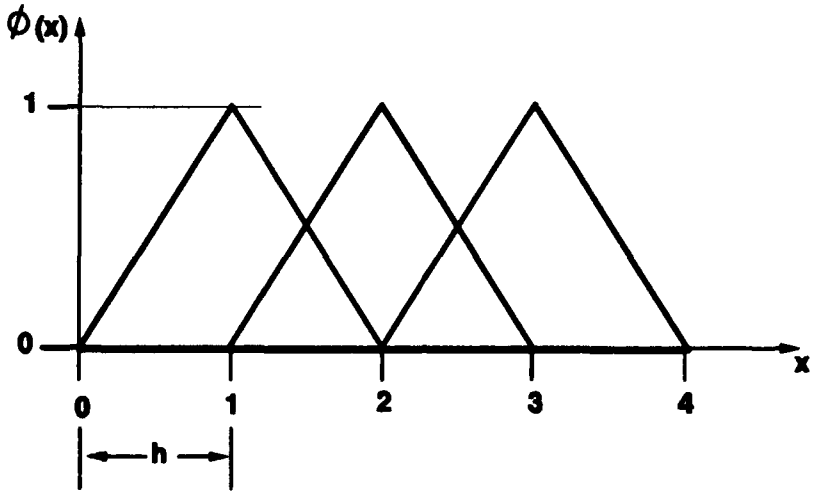


Figure 2.2
Finite Element Shape Functions

Numerical Integration

The integrations required in calculation of the solution to equation 2.18 can be accomplished analytically, but a numerical method of integration is preferred for lengthy computations. The programming of numerical solutions is an important step in allowing hundreds of integrals to be calculated over a finite element mesh. The main methods of numerical integration are the Newton-Cotes formulas, Romberg Integration, and Gaussian Quadrature.[2] The Newton-Cotes formulas are the simplest and most common integration schemes. Included in these are the Trapezoidal and Simpson's rules. Both the Simpson and Trapezoidal rule require that the domain of the problem be divided into bands. These bands are evaluated over the domain of the problem and each solution is summed to calculate the integral. The Newton-Cotes formulas are accurate if the band width being evaluated is small compared to the overall domain of the problem. As a

finite element mesh is refined, hundreds of elements are introduced into a problem. The time required to calculate a global solution will increase dramatically due to the number of calculations required on each element.

Romberg integration and Gaussian Quadrature can achieve a more accurate result than the Newton-Cotes formulas in fewer calculations. This will result in faster computation time. The methods are more complex than the Newton-Cotes formulas but can be easily adapted to computer code. Gaussian Quadrature is used to evaluate integrals in this program, and a discussion of the Gaussian Quadrature method follows.

Gaussian Quadrature

The following describes the derivation of the two point Gaussian Quadrature formula, this procedure can then be expanded to a higher number of integration points in the same manner. In this procedure, unlike the Newton-Cotes formulas, the nodal values being evaluated are not located at the end points of the element; their locations are considered unknowns.

Equation 2.20 is a quadrature formula which when evaluated gives the value of the integral of f(x) over the region. The equation has four unknowns and requires four equations to calculate the weighting functions c_1, c_2 and the location of the integration points, x_1, x_2. [2]

$$I = c_1 f(x_1) + c_2 f(x_2) \qquad 2.20$$

The solution of the four unknowns can be calculated if it is assumed the function fits a constant, linear, parabolic, and cubic term exactly. This is equivalent to the function being regarded as a general cubic polynomial, and results in equations 2.21 These four equations can be used to calculate the four unknowns. For convenience the integration is performed over the region from -1 to 1. [2]

$$c_1 f(x_1) + c_2 f(x_2) = \int_{-1}^{1} 1\,dx = 2 \qquad 2.21a$$

$$c_1 f(x_1) + c_2 f(x_2) = \int_{-1}^{1} x\,dx = 0 \qquad 2.21b$$

$$c_1 f(x_1) + c_2 f(x_2) = \int_{-1}^{1} x^2\,dx = \frac{2}{3} \qquad 2.21c$$

$$c_1 f(x_1) + c_2 f(x_2) = \int_{-1}^{1} x^3\,dx = 0 \qquad 2.21d$$

The simultaneous solution to the above set of equations results in the following:

$$c_1 = 1 \qquad x_1 = \frac{-1}{\sqrt{3}}$$

$$c_2 = 1 \qquad x_2 = \frac{1}{\sqrt{3}} \qquad 2.22$$

Values of c_1, c_2, x_1, and x_2 are substituted back into equation 2.20 resulting in equation 2.23.

$$I = 1*f\left(\frac{-1}{\sqrt{3}}\right) + 1*f\left(\frac{1}{\sqrt{3}}\right) \qquad 2.23$$

The solution to the integral over the domain of the problem can be quickly obtained by evaluating the integrand f(x) at points $x_1 = -1/\sqrt{3}$ and $x_2 = 1/\sqrt{3}$ and applying equation 2.23.

General Gaussian Quadrature formulation shown in equation 2.24, may be used to numerically integrate the expression for K_{ij} and F_i shown in equation 2.17 and 2.18.

$$\int_{-1}^{1} g(x)\,dx = \sum_{i=1}^{n} g(a_i)\,H_i \qquad 2.24$$

In the above equation, the integral of g(x) over the region -1 to 1 is evaluated. The index "n" is the number of Gaussian integration points, a_i is the x coordinate of the Gaussian point in a one dimensional problem, and H_i is the weighting function at the associated Gaussian Point. The formula allows numerical calculation of the integral by summing the corresponding values of the function evaluated at 'a_i' then multiplied by 'H_i' for each of the Gaussian Quadrature points.

The accuracy of the Gaussian Quadrature is dependent on the number of Gaussian integration points used and can be determined prior to the actual integration procedure. Gaussian Quadrature integrates a polynomial of order (2n-1) exactly, where n is the number of integration points. This can be applied for three Gaussian points to calculate the integral of a fifth order polynomial exactly. Additional values for weighting functions and integration points are given in table 2.1. [15]

Number of Gaussian Integration Points n	Generic Coordinate of Gaussian Integration Point $-1 < a_i < 1$ a_i	Weighting function associated with respective Gaussian Integration Point H_i
1	0.0	2.0
2	±0.5773502692	1.0
3	±0.7745966692 0.0	0.5555555556 0.8888888889
4	±0.8611363116 ±0.3399810436	0.3478548451 0.6521451549
5	±0.9061798459 ±0.5384693101 0.0	0.2369268851 0.4786286705 0.5688888889
6	±0.9324695142 ±0.6612093865 ±0.2386191861	0.1713244924 0.3607615730 0.4679139346

Table 2.1
Coefficients for Gaussian Quadrature

Isoparametric Elements

The integration points used in evaluating the integral over an element are specific to the element being evaluated. To increase the flexibility of the finite element method, the shape functions will be defined on a generic element. This will allow for the element in question to be transformed to the generic form, the integration performed, and then the result can be returned to the original coordinate system.

A general element is shown in figure 2.3, A and B are the element nodes in the local system. The coordinate system is represented by ξ, defined to be positive to the right and its origin is located at the left node.

The shape functions for the generic element are defined as shown below: [1]

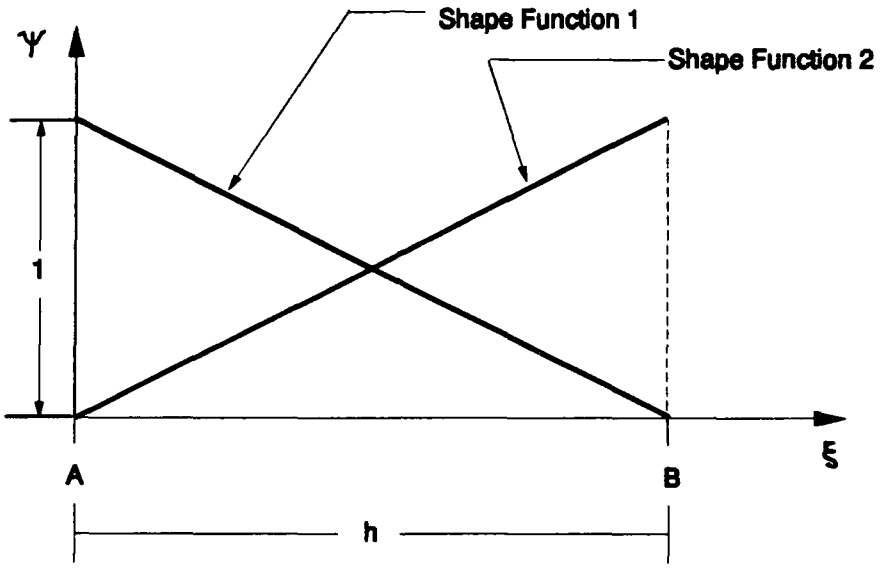


Figure 2.3
Generic Shape Functions

Shape function one is represented as follows:

$$\psi_A^e(\xi) = 1 - \frac{\xi}{h}$$
$$\psi_A^{e\prime}(\xi) = -\frac{1}{h} \qquad 2.25a$$

Shape function two is represented as follows:

$$\psi_B^e(\xi) = \frac{\xi}{h}$$
$$\psi_B^{e\prime}(\xi) = \frac{1}{h} \qquad 2.25b$$

The generic shape functions allows for the development of a system to integrate over elements that vary in size and location. The equations defining the shape functions, the development of the Galerkin approximation method, and the use of Gaussian Quadrature are all brought together to solve the second order boundary value problem.

General One Dimensional System

The ordinary linear second order equation is defined by equation 2.26.

$$\frac{d}{dx}\left[k(x)\frac{d}{dx}y(x)\right] + c(x)\frac{d}{dx}y(x) + b(x)y(x) = f(x) \qquad 2.26$$

- **k(x)** $\neq$ **0** and does not change sign
- **y(x)** is a variable to be solved for

In this one dimensional case (2.26) the coefficients k, c, b, and f are functions of x defined over the given domain. Let the following represent the variables in the above equation.

$$\frac{d}{dx}\,y(x) = y' \qquad \frac{d^2}{dx}\,y(x) = y'' \qquad 2.27$$

The equation can now be rewritten into a simpler form as shown in equation 2.28.

$$-K(x)\,y'' + c(x)\,y' + b(x)\,y = f(x) \qquad 2.28$$

Equation 2.28 can also be written to a variational form as shown in equation 2.29.

$$\int_0^L (\bar{y}\frac{d}{dx}(ky') + c\bar{y}y' + b\bar{y}y) = \int_0^L \bar{y}\, f dx \qquad 2.29$$

Where $\bar{y}$ is a test variable like the variable v used in the model problem.

Equation 2.30a and 2.30b are determined by integrating the first term of equation 2.29 by parts.

$$\int_0^L (k\,\bar{y}'\,y' + c\bar{y}\,y' + b\bar{y}y)\,dx - k\,\bar{y}y'\,|_0^L = \int_0^L \bar{y}\, f dx \qquad 2.30a$$

and

$$\int_0^L (k\,\bar{y}'\,y' + c\bar{y}\,y' + b\bar{y}\,y)\,dx = \int_0^L \bar{y}\, f dx + k\,\bar{y}y'\,|_0^L \qquad 2.30b$$

Evaluating the above results in equation 2.31

$$\int_0^L (k\bar{y}y' + c\bar{y}y' + b\bar{y}y)\,dx = \int_0^L \bar{y} f dx + \bar{y}(L)\,k(L)\,y'(L) - \bar{y}(0)\,k(0)\,y'(0) \qquad 2.31$$

Similar to the model problem, domain basis functions are represented by element shape functions and are introduced for y, y', $\bar{y}$, and $\bar{y}'$.

$$y = \lfloor \psi(x) \rfloor \{y_i\} \qquad y' = \lfloor \psi_j'(x) \rfloor \{y_i\} \qquad 2.32a$$

$$\bar{y} = \lfloor \bar{y}_j \rfloor \{\psi_i(x)\} \qquad \bar{y}' = \lfloor \bar{y}_j \rfloor \{\psi_i'(x)\} \qquad 2.32b$$

Substitution of equations 2.32a and 2.32b into the variational form (2.31) results in equations 2.33.

$$\lfloor \overline{y}_j \rfloor \; [K_{ij}] \; \{y_i\} = \lfloor \overline{y}_j \rfloor \{F_i\} + \lfloor \overline{y}_j \rfloor \{k(L)\, y'(L), \ldots, -k(0)\, y'(0)\} \qquad 2.33$$

by removing the test variable $\overline{y}_j$ equation 2.34 is formed

$$[K_{ij}]\{y_i\} = \{F_i\} + \{k(L)\, y'(L), \ldots, -k(0)\, y'(0)\} \qquad 2.34$$

Equation 2.35 is obtained by redefining F_i to incorperate the last term

$$[K_{i\,j}]\,[y_i] = [F_i] \qquad 2.35$$

where:

$$[K_{i\,j}] = \left[\int_0^L (\psi_i' \psi_j' + c\, \psi_i \psi_j' + b\, \psi_i \psi_j)\, dx \right] \qquad 2.36a$$

and

$$\{F_i\} = \left\{ \int_0^L \psi_i\, f dx \right\} + \{k(L)\, y'(L), \ldots, -k(0)\, y'(0)\} \qquad 2.36b$$

- Finite element shape functions are represented by ψ_i

The last term of equation 2.36b represents the natural boundary conditions. This term is added at x = 0 and x = L, where 0 and L are the end points of the domain for this one dimensional problem. If y', the rate of change with respect to distance, is zero at the boundaries then the term is omitted and only essential boundary conditions are

applied. Note that the natural boundary conditions do not apply on the interior of the domain. After performing the above calculations a program can be developed to compute solutions to these equations.

Two dimensional problem

The general two dimensional boundary value problem is described below, along with an approach to develop a numerical solution for the two dimension boundary value problem.

$$-\nabla\cdot[k(x,y)\,\nabla u(x,y)\,]+c(x,y)\left[\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}\right]+b(x,y)\,u(x,y)=f(x,y) \qquad 2.37a$$

The same approach is taken with the two dimensional problem as was taken with the model problem and the one dimensional problem. Employing a test variable and integrating over the domain results in equation 2.37b.

$$\int_x\int_y\{-\nabla\cdot[k(x,y)\,\nabla u(x,y)\,]+c(x,y)\left[\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}\right]+$$

$$b(x,y)\,u(x,y)\}\bar{u}\;dydx=\int_x\int_y f(x,y)\;\bar{u}\;dydx \qquad 2.37b$$

Green's Theorem for two dimensions allows for double integration over the domain of a problem to be transformed to a line integral over the boundary of the problem.

Greens Theorem is shown below.

$$\iint_{x\,y}\left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y}\right)dxdy = \int_{\Omega}(u\,dx + u\,dy)\,dxdy \qquad 2.38$$

- u is the variable being solved for
- Ω is the boundary defining the domain of the problem

This can be rewritten for the Laplacian shown in equation 2.39 as,

$$\iint_{R}\nabla^2\,u\,dxdy = \int_{\Omega}\frac{\partial u}{\partial n}ds \qquad 2.39$$

- R is the domain of the problem
- Ω is the boundary defining the domain of the problem

The mathematics describing Green's theorem in detail is contained in reference [8], and the application of Green's theorem to finite elements is explained in reference [1]. It, in effect, allows the integration by parts in two dimensions of the second order term in equation 2.36b.

Integration by parts gives us the variational form below:

$$\iint_{x\,y}(k\,\nabla\overline{u}\cdot\nabla u + c\,\overline{u}\nabla u + b\,\overline{u}u)\,ds = \iint_{x\,y}\overline{u}f\,ds + \iint_{x\,y}u(s)\,k(s)\,\frac{\partial u(s)}{\partial u}ds \qquad 2.40$$

The state variable and test functions are defined below in terms of element shape functions:

$$u(x,y) = \lfloor \psi_j(x,y) \rfloor \{ u_i \} \qquad 2.41a$$

$$\overline{u}(x,y) = \lfloor \overline{u}_j \rfloor \{ \psi_i(x,y) \} \qquad 2.41b$$

and

$$\nabla u = \lfloor \nabla\psi_j(x,y) \rfloor \{ u_i \} \qquad 2.42a$$

$$\nabla\overline{u} = \lfloor \overline{u}_j \rfloor \{ \nabla\psi_i(x,y) \} \qquad 2.42b$$

Substituting these forms and their derivatives into equation 2.38 results in the following.

$$[K_{ij}] \{u_i\} = \{F_i\} + \int_{2\Omega} \left[k(s) \frac{\partial}{\partial n} u(s) \psi_i(s) \right] ds \qquad 2.43$$

where:

$$K_{ij} = \int_{\Omega} (k \nabla\psi_i \cdot \nabla\psi_j + c \psi_i \nabla\psi_j + b \psi_i \psi_j) ds \qquad 2.44$$

or

$$K_{ij} = \int_y \int_x \left\{ k(x,y) \left[\frac{\partial\psi_i}{\partial x} \frac{\partial\psi_j}{\partial x} + \frac{\partial\psi_i}{\partial y} \frac{\partial\psi_j}{\partial y} \right] + c \left[\psi_i \left(\frac{\partial\psi_j}{\partial x} + \frac{\partial\psi_j}{\partial y} \right) \right] + b(\psi_i \psi_j) \right\} dx dy \qquad 2.45$$

Where:

$$\nabla\psi(x,y) = \frac{\partial}{\partial x}\psi(x,y) + \frac{\partial}{\partial y}\psi(x,y) \qquad 2.46$$

and finally

$$F_i = \iint_{yx}[f(x,y)\,\psi_i]dxdy + \int_{2\Omega}\left[k(s)\,\frac{\partial u(s)}{\partial n}\psi_i(s)\right]ds \qquad 2.47$$

The equation set from above can be evaluated in the same manner as those in the one dimensional case. The only differences are the shape functions used in the evaluation of the integral and the integration, both of which involve two independent variables.

Shape Functions

The shape functions used in the program are square quadrilateral elements. Triangular elements also adapt well to the finite element mesh but were not employed. These generic quadrilateral elements will be shown below and on the following page.

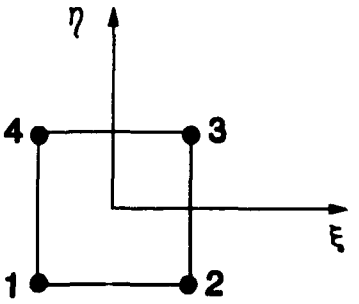


Figure 2.4
Quadrilateral Bilinear Element

Shape Functions

$$\psi_1 = \frac{1}{4}(1-\xi)(1-\eta)$$
$$\psi_2 = \frac{1}{4}(1+\xi)(1-\eta)$$
$$\psi_3 = \frac{1}{4}(1+\xi)(1+\eta)$$
$$\psi_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

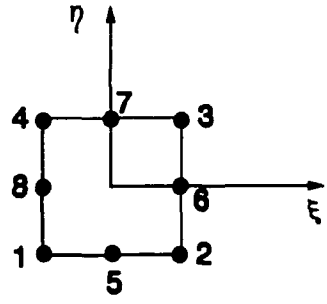


Figure 2.5
Quadrilateral Quadratic
Serendipity Element

Shape Functions

$$\psi_1 = \frac{1}{4}(1-\xi)(1-\eta)(-1-\xi-\eta)$$
$$\psi_2 = \frac{1}{4}(1+\xi)(1-\eta)(-1+\xi-\eta)$$
$$\psi_3 = \frac{1}{4}(1+\xi)(1+\eta)(-1+\xi+\eta)$$
$$\psi_4 = \frac{1}{4}(1-\xi)(1+\eta)(-1-\xi+\eta)$$
$$\psi_5 = \frac{1}{2}(1-\xi^2)(1-\eta)$$
$$\psi_6 = \frac{1}{2}(1+\xi)(1-\eta^2)$$
$$\psi_7 = \frac{1}{2}(1-\xi^2)(1+\eta)$$
$$\psi_8 = \frac{1}{2}(1-\xi)(1-\eta^2)$$

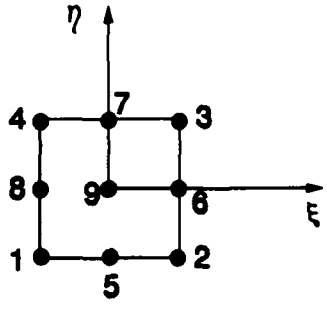


Figure 2.6
Nine Node Quadrilateral
Quadratic Element

Shape Functions

$$\psi_1 = \frac{1}{4}(\xi^2-\xi)(\eta^2-\eta)$$
$$\psi_2 = \frac{1}{4}(\xi^2+\xi)(\eta^2-\eta)$$
$$\psi_3 = \frac{1}{4}(\xi^2+\xi)(\eta^2+\eta)$$
$$\psi_4 = \frac{1}{4}(\xi^2-\xi)(\eta^2+\eta)$$
$$\psi_5 = \frac{1}{2}(1-\xi^2)(\eta^2+\eta)$$
$$\psi_6 = \frac{1}{2}(\xi^2+\xi)(1-\eta^2)$$
$$\psi_7 = \frac{1}{2}(1-\xi^2)(\eta^2+\eta)$$
$$\psi_8 = \frac{1}{2}(\xi^2-\xi)(1-\eta^2)$$
$$\psi_9 = 1(1-\xi^2)(1-\eta^2)$$

In the equations on the previous two pages, the length of any side of an element is equal to two. The center of the element is located at the origin of the ξ, η grid defining the general element. By defining a general element, non-square elements can be transformed into a form that allows for integration to be carried out simply. Once the integration is completed the transformation can be reversed to return these values to the original coordinate system.

CHAPTER 3

OVERVIEW OF THE PROGRAM

The program developed in this paper is a continuation of the program LAPLACE.FOR by Dr. W. F. Carroll, Professor, University of Central Florida - D/CEE. The program Dr. Carroll developed was capable of solving a second order boundary value problem in the form of Laplace's Equation. This program was used as the basis for the program developed in this paper. The program LAPLACE.FOR was written in FORTRAN and had to be re-coded into Microsoft Basic in order to facilitate the programming of graphics to represent the solution to the problem being evaluated. The program was modified and the following features added;

1. Availability to store multiple data files, data files that can hold previously entered problems.
2. A more generalized problem domain entry procedure, and the generation of a graphic representation of the mesh in the domain.
3. Calculation of equipotential lines and the ability to store the data in a user designated file, and the graphical presentation of them.
4. Calculation of flow lines and storage in a user designated file, and the graphical presentation of them.

These changes were made to improve the usefulness of the program and give the user a better understanding of the results calculated by the program. Both the selection and adaptation of equipotential lines and flow lines are done interactively by the user to achieve the flow net desired.

LAPLACE was developed to calculate the solution to a second order two dimensional partial differential equation. The version of the program included here is designed to solve a specific second order boundary value problem, Laplace's Equation. The intent in the development of the program was not to solve Laplace's Equation uniquely, but develop a universal program that could be easily modified to solve differing types of second order equations and represent the solutions graphically. This can be accomplished in the program LAPLACE with only slight changes to the programming code localized in one of the subroutines.

Program Development

The program allows for the problem domain to be entered by regions or, in the terminology of program LAPLACE, subdomains. These subdomains are subdivided into elements and then stored in a data file specified by the user. This data file can be recalled and different boundary conditions applied to a previously entered file for any subsequent running of a similar problem with the same domain.

Boundary conditions are entered following a display of the program developed mesh. Up to 25 boundary conditions can be applied to a problem; these can be essential, natural, or a combination of essential and natural boundary conditions. The boundary

conditions are written to the data file containing the subdomain data. After the last boundary condition is entered, the program will execute a series of subroutines and determine a solution for the boundary value problem. The solution values of the state variable for the problem will be assigned to each node in the developed mesh. These values are also written to the data file containing the boundary values and subdomain data.

After the solution to the problem is calculated the user can view the results in a graphical format. The solution to Laplace's Equation is developed further to determine pressure gradients across the domain. The development of pressure gradients allows for computation and graphical display of contour lines and lines perpendicular the contour lines. The contour lines represent lines of equal pressure and their perpendiculars represent flow lines.

Program Requirements

The Program LAPLACE is written in 'basic' computer language and compiled to be run under DOS 5.0. The program was compiled using Microsoft Basic Compiler version 7.0 and can run on a personal computer with or without a math coprocessor. If a coprocessor is present the program will utilize it.

Graphics can be printed by using the GRAPHICS command supplied under DOS 5.0, prior to execution of the program LAPLACE. The GRAPHICS command allows flow nets displayed on the screen by the program LAPLACE to be printed by a local printer. After GRAPHICS is run, the Shift-Print-Screen command will print the graphics

displayed on the screen to a local printer. Table 3.1 lists additional DOS 5.0 commands that enable the print-screen option when using a printer other than an IBM graphics printer or compatible. These commands are used in the same fashion as the GRAPHICS command explained above. [9]

DOS 5.0 Command	Local Printer Type
COLOR1	IBM Personal Computer color printer with black ribbon
COLOR4	IBM Personal Computer color printer with RGB ribbon
COLOR8	IBM Personal Computer color printer with CMY Ribbon
HPDEFAULT	Any Hewlett-Packard PCL printer
DESKJET	Hewlett-Packard Deskjet Printer
GRAPHICS	IBM Personal Graphics Printer or compatible
GRAPHICSWIDE	IBM Personal Graphics Printer or compatible with 11 inch carriage
LASERJET	Hewlett-Packard LaserJet Printer
PAINTJET	Hewlett-Packard PaintJet Printer

Table 3.1
Dos 5.0 Commands for Graphics Printing

Data Entry

The subroutine ENDAT, within program LAPLACE, asks for the data used to develop the mesh. The data is entered by rows of subdomains from bottom to top. The program requires subdomains to be entered in a continuous fashion from the first to the last sub domain. A maximum of sixteen subdomains are allowed to represent the entire domain of the problem. For each row of subdomains, the location of the lower left node of the first active subdomain must be entered.

The coordinates of the lower-left node is with respect to the mesh to be generated by the program LAPLACE, not the cartesian coordinates of these nodes. The coordinates of the subdomain location node are determined in the following fashion: The row of subdomains with the left most sub domain will have a location value of 1 for the lower left node of the first subdomain in that row. The coordinates for the lower left position of following rows will increase by the number elements of the first subdomain in the subsequent row is displaced to the right. After the coordinates are entered for all rows, the program will compute a finite element mesh and use it to calculate the solution to the boundary value problem. The mesh will be displayed giving the user a visual representation of elements, nodes, and the coordinate system. This allows the user to see if the mesh is entered correctly and gives a better understanding of the numbering sequence of the elements. The element numbers need to be known to enter boundary conditions.

Mesh data is correctable after the last value for each subdomain is entered or during the review of all the subdomain data after all is entered. The mesh data is then

placed in a file on the users disk and the user can call this data file in a subsequent running of the program. At the time the mesh is presented graphically on the screen, a decision can be made to continue or stop the program. If the program is stopped, execution of the command line LAPLACE is required to reboot the program to reset all the arrays.

Data for each subdomain is entered according to the following guidelines for the program to operate properly:

1. Subdomains are consecutively numbered from left to right, bottom to top.
2. No open areas can be introduced between sub-domains in a particular row of subdomains.
3. All subdomains in a specific row have to have the same corresponding number of rows of elements.
4. The nodal coordinates and number of elements located along subdomain boundaries have to align with any other common subdomain boundary.
5. The maximum number of sub-domains is sixteen.
6. The furthest left subdomain will have a location value of one for its first column of nodal coordinates.

Mesh Development

The program LAPLACE uses a mesh generator to aid in the entering of problem domain data. The problem can be subdivided into smaller areas, called subdomains, instead of entering each nodal coordinate for the entire domain of the problem. These subdomains can then be specified to have a certain number of rows of elements with specific number of elements in each row. This procedure of mesh generation quickens the input of the domain data and allows the problem to be subdivided into smaller elements with the program calculating nodal coordinates for these smaller elements.

Boundary Conditions

The subroutine BCDAT of program LAPLACE allows the user to enter boundary conditions for the mesh previously displayed. The boundary conditions entered can be essential, natural, or a combination of the two. The number of boundary conditions must be specified. A single boundary condition may be applied continuously along a strip of elements on the boundary of the domain. After the number of boundary conditions (boundary strips) has been entered, the following format and sequence is required when entering each of the boundary conditions (at a strip).

1st - Type of boundary condition, enter

1 - for essential,

2 - for natural, or

3 - for mixed

2nd - Enter the numerical value for the boundary condition.

The numerical value entered for a boundary condition is dependent on the boundary condition specified in step 1. The following applies to the boundary conditions entered in this step.

a. For essential boundary conditions a value of the state variable is entered.
b. For natural boundary conditions a value of the normal derivative of the state variable to the boundary is entered.
c. For a mixed boundary condition each of the above is entered with the essential boundary condition entered first followed by the natural boundary condition.

3rd - Enter the side of the element that the boundary condition is applied to.
1 - for bottom,
2 - for right,
3 - for top, or
4 - for left

4th - Enter the number of the first element that the boundary condition applies to.

5th - Enter the last element number of the associated boundary condition.

6th - Enter the increment of element numbers between elements that this boundary condition applies.

Boundary condition data can be corrected by the user after the last boundary condition for a strip has been entered. Correction is again allowed during the review of all the boundary condition data. This data is also written to a user defined file with the subdomain data.

Data Computation and File Designation

The program next calculates the values of the state variable at the nodal coordinates of the mesh, and uses this data to calculate and draw the contour and flow lines and graphically represents the solution to the problem. The number of contours and flow lines can be varied, allowing the user to achieve the best graphical solution for their problem.

CHAPTER 4

EXAMPLE PROBLEM

Problem Statement

Calculate the internal pressure variation in the domain shown in figure 4.1. Draw contour and flow lines to represent the solution in a graphic manner. Essential boundary conditions are given as the pressure along the lower left edge at 15 psi and along the upper right edge at 30 psi. Flow is prohibited from crossing boundaries anywhere else. Dimensions of the domain are as shown in figure 4.1.

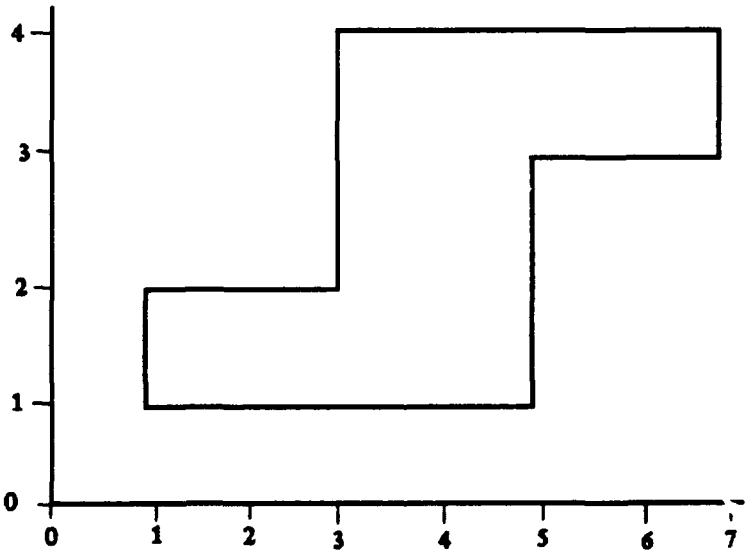


Figure 4.1
Example Problem Domain

The problem can be divided into numerous arrangements of subdomains depending on the needs of the program user. A finer mesh can be used in specified subdomains to increase the number of nodal values calculated for that subdomain. The criteria for subdomain partitioning from chapter three is still observed when dividing the problem domain into subdomains.

The problem could be solved by dividing the domain into three subdomains as shown in figure 4.2.

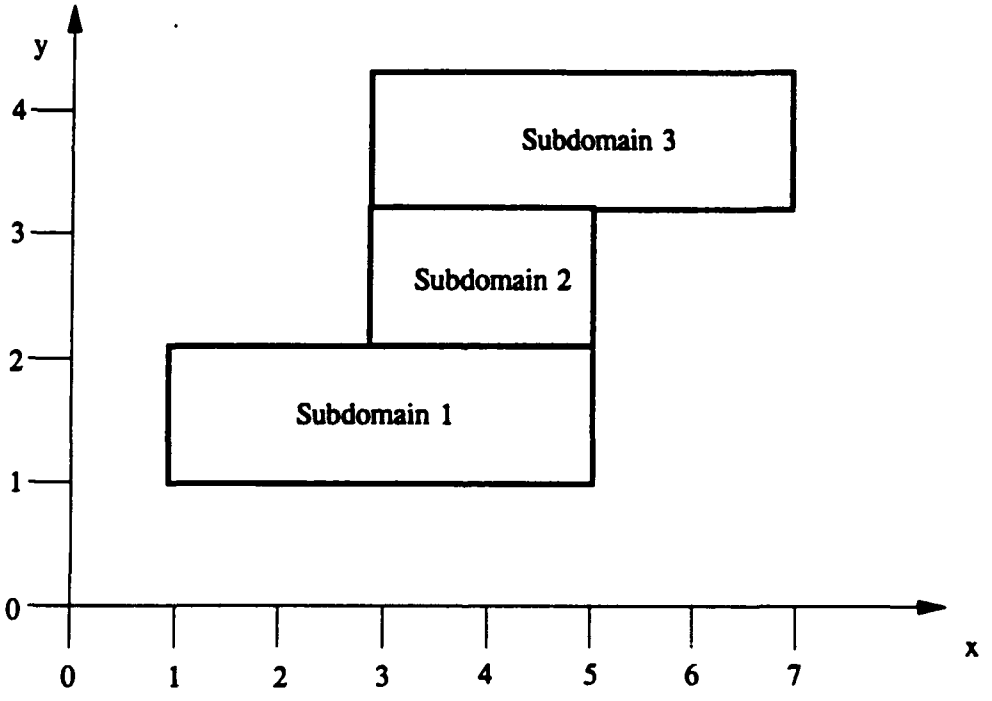


Figure 4.2
Domain Separated into Three Subdomains

The solution to the example problem located in appendix B was obtained by dividing the domain of the problem into five subdomains. This division allows the mesh to be finer in the center portion of the domain than at the ends. Each subdomain has four nodal values. The sketch below shows the position of the five subdomains and how each of the subdomains are broken into a number of elements. The position of the lower left node for the first active subdomain in each row is also shown in the sketch.

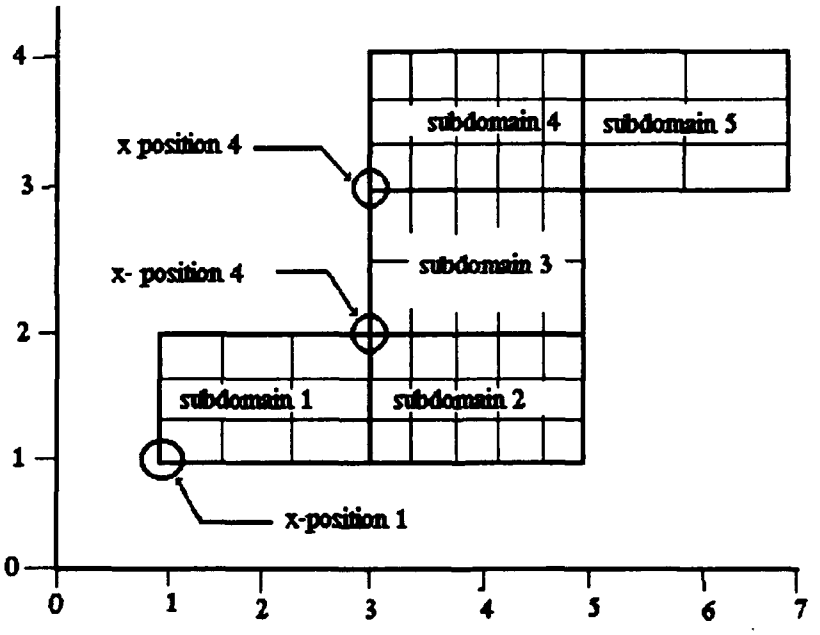


Figure 4.3
Subdomain Positioning

The number of elements per row and number of rows of elements for each sub-domain is listed below:

Sub-domain 1 -- 3 rows of elements -- 3 elements in each row

Sub-domain 2 -- 3 rows of elements -- 5 elements in each row

Sub-domain 3 -- 2 rows of elements -- 5 elements in each row

Sub-domain 4 -- 3 rows of elements -- 5 elements in each row

Sub-domain 5 -- 3 rows of elements -- 2 elements in each row

The following procedure should be followed to obtain the solution in appendix B for the problem described above.

Preliminary Data Entry

1. Type LAPLACE and press enter (or return) to start program running and a title screen will appear, press enter to continue.
2. The program inquires if an existing data file will be used. Type "n" and press return.
3. Next a name has to be entered that will designate the data that will identify this problem. EXAMPLE is used as the problem name for the solution shown in Appendix B. Type "Example" and press return.

Domain Entry

1. The program first requires the number of rows of subdomains. Example has 3 rows of subdomains, enter "3" and press return.
2. The number of subdomains in each row of subdomains is entered next. Example has 2 sub-domains in row 1, 1 in row 2 and 2 in row 3, working from bottom to top. Enter "2" for the first row of sub-domains and press return. The program now requires the position of the lower left hand node of

the first active subdomain in row number 1. This position is determined by the size of the mesh and the left most edge of the mesh determines position 1 in the mesh. Because subdomain 1 is the left most subdomain in the sample problem, the position of the lower left corner of sub-domain 1 is 1. Enter "1" and press return.

3. The process is repeated for each row of subdomains. Row two has 1 sub-domain and the position in the mesh of the lower left corner of sub-domain number three is 4. This is determined by the number of elements in each row of Subdomain #1. Subdomain #1 has three elements, determining the position of all sub-domains above. Note, if sub-domain 1 had 6 elements in each row than the position of the lower left node for sub-domain #3 would be 7 and so on. Enter 1, return, 4, return for row two and 2, return, 4, return for row three.
4. The program will next ask for the number of integration points and nodes associated with the element of each subdomain. Our example will use four integration points and four nodal points for each subdomain. Enter 4, return, 4, return.
5. Correction of the data entered above is now allowed. If all data are entered correctly, press 'y', return; if not press 'n', return and repeat steps 1 through 4 above.

The program LAPLACE requires additional data to build the finite element mesh necessary to solve the problem. The following steps specify how data should be

entered for each of the sub-domains to correctly allow the program to develop a suitable mesh.

The program will prompt for the number of rows of elements in a subdomain #1 the corresponding number of elements per row for subdomains #1 and key nodal coordinates of subdomain #1. The key nodes of the subdomain correspond in position to the nodes on the type of generic element being used. This procedure is repeated for each subdomain, in succession, starting from subdomain #1.

1. The first subdomain of the example problem has 3 rows of elements with 3 elements in each row: enter 3, return, 3, return.
2. Next the program then requires the user to specify the cartesian coordinates for each of the key nodes of the subdomain being entered. The example program has 4 nodal coordinate values that need to be entered for subdomain #1, since four node quadrilateral elements are being used. These values are entered starting at the lower left corner of the subdomain and continuing around the subdomain in a counterclockwise fashion entering the x, and y coordinate for each node. The following would be entered for sub-domain #1: '1,1,3,1,3,2,1,2'. The return key must be pressed after each value.
3. The program then allows the user to correct this data after the last coordinate value is entered for the subdomain in question. If all data is correct, enter y, return and the program will continue to the next subdomain. If a correction is necessary enter n, return and steps 1 & 2 can be repeated for the last subdomain.

4. Enter the following data for the remaining four subdomains

Subdomain #2

3 - rows of elements, 5 - Elements per row

	Node	1	2	3	4
coordinates	x	3	5	5	3
	y	1	1	2	2

Subdomain #3

2 - rows of elements, 5 - Elements per row

	Node	1	2	3	4
coordinates	x	3	5	5	3
	y	2	2	3	3

Subdomain #4

3 - rows of elements, 5 - Elements per row

	Node	1	2	3	4
coordinates	x	3	5	5	3
	y	3	3	4	4

Subdomain #5

2 - rows of elements, 5 - Elements per row

	Node	1	2	3	4
coordinates	x	5	7	7	5
	y	3	3	4	4

5. The data can be reviewed and reentered, if desired, after the final subdomain has been entered. Enter 'y, return' to review the data and make any necessary corrections. After all corrections are made, press return and the program will display a graphical representation of the mesh developed for this problem. Compare the mesh on the screen with that shown top of next page.

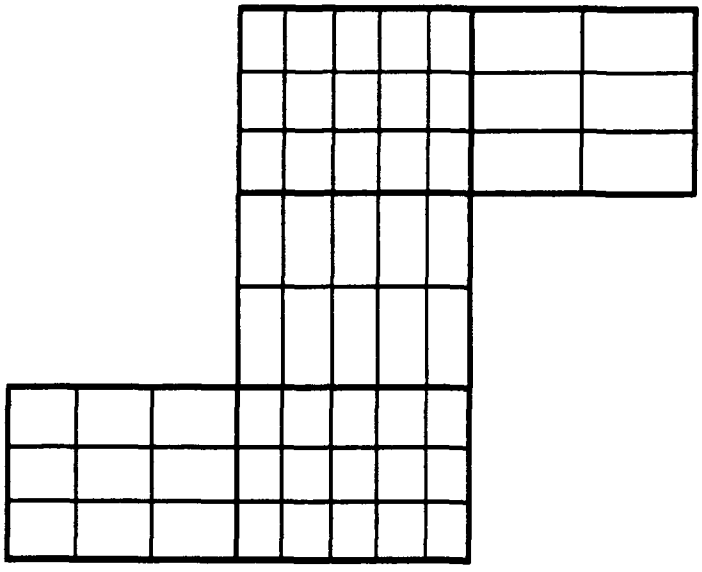


Figure 4.4
Element Layout of Subdomains

If your mesh does not resemble the mesh shown above then press "n" , restart Laplace.exe and reenter the above data to achieve this mesh.

The program Laplace develops the mesh shown above and numbers the elements in a consecutive manner from left to right starting at the bottom and working up toward the top of the page. The numbering scheme for the mesh developed here is in figure 4.5. Fifty five elements make up the mesh in the example problem. This arrangement will be used to enter the boundary conditions required to solve the problem.

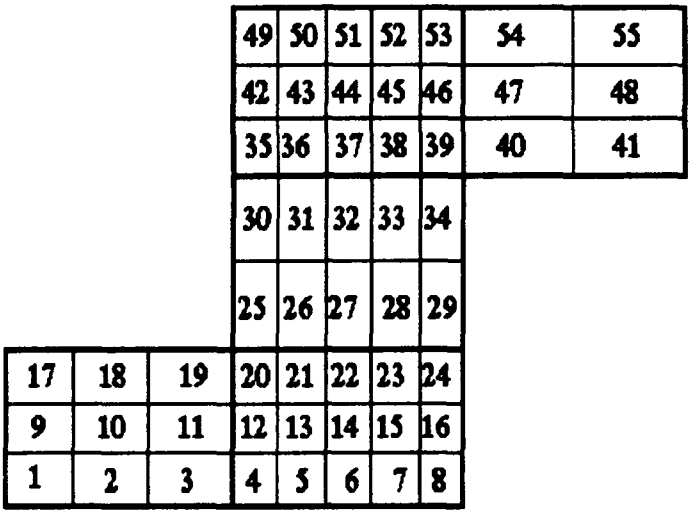


Figure 4.5
Element Numbering Sequence

Boundary Condition Entry

Up to 25 boundary strips for boundary conditions may be entered in the following manner:

a. The types of boundary condition, enter:

1 - for essential
2 - for natural
3 - for mixed (combination of essential and natural)

b. The value of the essential, natural, or mixed boundary condition.

c. The side of the element on which the boundary condition is applied.

1 - for bottom
2 - for right side
3 - for top
4 - for left side

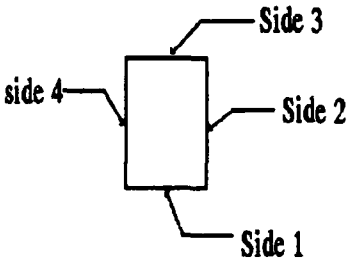


Figure 4.6
Element Face Boundary Conditions

d. The first element number that the boundary condition is applied to.

e. The last element number that the boundary condition is applied to.

f. The increment of element numbers between successive elements to which the boundary condition applies.

In the example problem, we know two essential boundary conditions: the first is at the left face of the domain from coordinate (1,1) to (1,2); the other is at the right face of the domain from coordinate (7,3) to (7,4). The value of the essential boundary condition at surface one is equal to 15 psi, and at surface two is 30 psi. We also know all other boundary have a natural boundary condition of zero.

The following illustrates the procedure for entering boundary conditions on the left and right boundaries stated above.

1. Enter '2' for the number of boundary strips, and press return.
2. Enter the following 6 values for the next series of questions for boundary conditions on the left side.

1 - Essential boundary condition

15 - Value of essential boundary condition

4 - Left face of element #1

1 - The first element associated with this boundary condition

17 - The last element associated with this boundary condition

8 - Increment between elements on the left face where boundary condition is applied

After the values for boundary condition #1 are entered, the program will ask for the data values for the second boundary condition. The second set of boundary conditions are as follows.

1 - Essential boundary condition

30 - Value of the essential boundary condition

2 - The right face of the 41st element

41 - The first element effected by the boundary condition

55 - The last element associated with the second boundary condition

7 - The increment between elements on the left face

Essential boundary conditions and natural boundary conditions that are different from zero are the only conditions that need to be entered. If a boundary condition is not entered it is assumed by default to be a natural boundary condition equal zero, therefore flow can only occur where essential boundary conditions or natural conditions not equal to zero are specified on the boundary of the problem. A minimum of two boundary conditions are required to calculate a solution to the differential equation and solve the problem.

After all the boundary conditions have been entered, the program allows any corrections necessary by boundary condition number. If the boundary conditions are not corrected at this time, they can still be corrected during the review and display after entry of all the boundary condition data.

Once all the boundary condition data is entered the program is ready to run. LAPLACE will assemble and band the global stiffness matrix and calculate the solution values for the finite element mesh. The solution values for the nodal points along with the subdomain data and the boundary condition data will be written to a file which was named earlier. It will be given the three letter extension ".MSH".

Equipotential Lines

The program next calculates the equipotential lines from the values calculated at the nodal points of the mesh. The program asks for the number of equipotential lines to draw across the domain of the problem. These lines can be drawn over the background grid of the mesh or just in the boundaries of the problem with the mesh omitted. After the equipotential lines are drawn, the user can continue to redraw the lines with different numbers of equipotential lines until an interval is found that best represents the solution to the problem. Figure 4.7, located at the top of the next page, shows the solution employing fifty contour lines. This is a general solution to aid the user in using the program and the number of contour lines can be varied and will effect the flow line calculation.

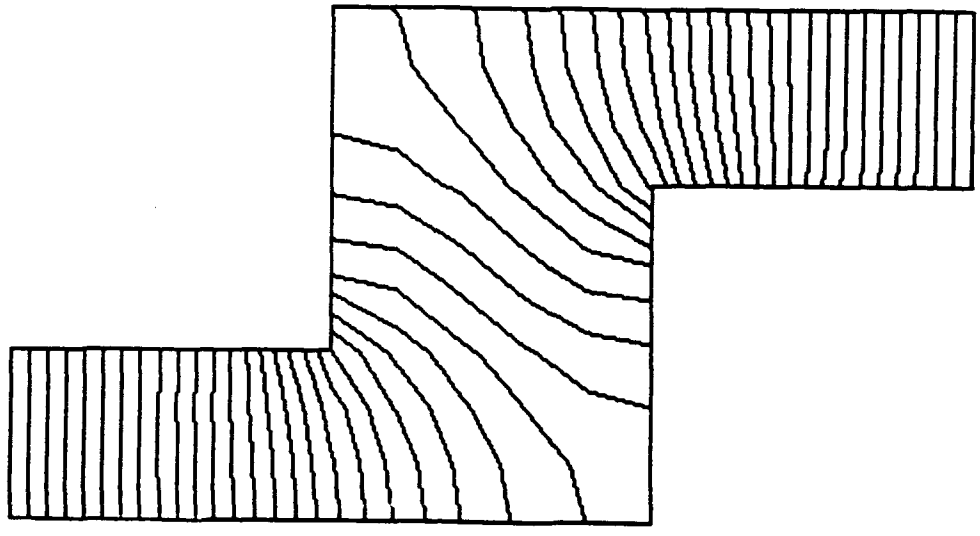


Figure 4.7
Contour Graph

The coordinates for the contour lines are sorted to align data points along the contour in a linear fashion and written into a file with the project name, followed by the three letter identifier ".CTR". A screen displaying "contour sorting" will be displayed until sorting is complete and all data has been written to a file. The sorting of contour data is required for flow lines to be calculated properly.

Flow Net Generation

The program uses the last equipotential interval displayed to compute the perpendicular components that will make up the solution net. Here the program requires the number of flow lines desired to be generated across the domain of the problem. The spacing of the contour lines can also be varied at this time to allow the development of a net that is easy to read. Figure 4.8 shows the solution with five flow lines employed.

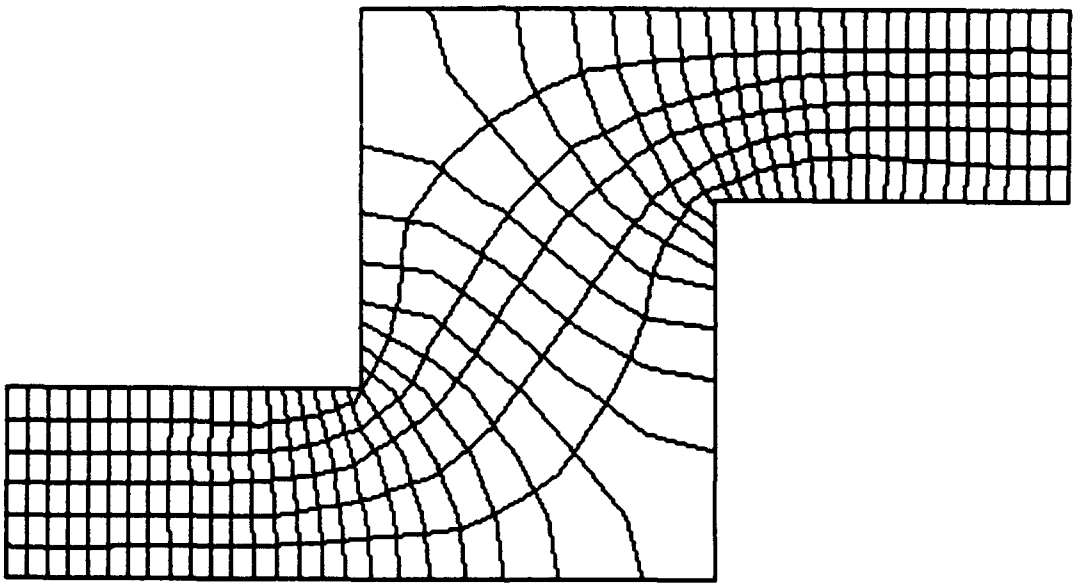


Figure 4.8
Flow Line Graph

The net developed by the equipotential lines and their perpendiculars can be plotted over the original mesh or just over the domain of the problem with the mesh omitted. After the users last version of his net has been generated, the program will save this data to a file specified with the project name and the three letter extension ".FLO".

The program will have to be restarted to reset any large array that may have been used in the previous running of the program.

See appendix B for the actual output developed by the example problem just described. The data in the appendix was developed using fifty equipotential lines and five flow lines over the domain of the problem.

Data is stored in the following files -

"EXAMPLE".DAT -- Data used to develop domain

"EXAMPLE".MSH -- Solution to problem at the nodal points

"EXAMPLE".CTR --- Contour data points

"EXAMPLE".FLO --- Flow line data points

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The development of the program LAPLACE required an understanding of matrix manipulation and programming language, along with the topics discussed earlier in this paper. Ten conclusions are presented below summing the development and running of the program.

1. The ability to design a program that will run an analysis of a two dimensional boundary value problem requires a good understanding of the finite element method and the application of Galerkin Approximations.
2. Program LAPLACE can be used as a tool in the solution of differing types of engineering problems, not only flow problems defined by Laplace's Equation. It was specifically validated for Laplace's Equation.
3. The program was successful in graphically displaying the solution of two dimensional second order boundary value problem over the domain of the problem. The program calculated nodal values for the problem, and used these values to generate equipotential lines and their perpendiculars. The graphical display produced on the screen could be easily interpreted and gives the user a visual solution to the problem.

4. The time taken to solve the example problem was reasonable (2-5 min). Time for such a problem solution depends on the computer used. Calculation of a larger problem requires increased computation time. This time can dramatically increase with the introduction of additional rows and columns of elements in subdomains.
5. The flow nets drawn to the screen become more accurate as the finite element mesh is refined. There is a tradeoff between calculation time and the accuracy of the flow net calculated. The finer the finite element mesh and the greater the number of nodal values being calculated, the more realistic the contour values.
6. The accuracy of the equipotential lines and their perpendiculars calculated in specific region on the domain can be increased by refining the mesh in the area of question. By using a uniform grid spacing on the first run of the program the graphical output will appear distorted at discontinuities in the system. These discontinuities can be corrected by refining the mesh at the locations in question. The refinement of the mesh will increase the number of nodal points calculated at the discontinuity and directly increase the refinement of the equipotential lines and their perpendiculars in that region. The program also allows the user to increase the number of contour lines used to calculate the flow net. A greater number of contour lines results in a more refined flow net but also increases computation time.
7. The largest limitation of the program is the inability for Microsoft Basic Language 7.0 to directly access the extended memory that is available on many personal computers. The program runs out of memory in the calculation of large finite element meshes, this varies with the arrangement of the mesh and the division of the

subdomains. If the extended memory of the personal computers could be accessed the size of the finite element mesh being solved could be increased. The newer versions of Microsoft Fortran and Microsoft Pascal have procedures built into the compiling process that gives programs written in these specific languages access to extended memory. Future versions of BASIC may have the capability to compile existing basic programs to use extended memory with little or no change in the processing code. The access to extended memory would give this program added value and increase the refinement of finite element meshes that could be developed.

9. The graphic code of program LAPLACE draws contour and flow lines for data written to a data file not internal to the finite element calculations. The graphics subroutines of this program could be manipulated to read data from a file generated by the external program if a different finite element program was used to obtain the values of state variables at nodes in a mesh. This would give the user a graphical representation of the solution to the two dimensional problem developed outside of the program LAPLACE.EXE.
10. The program LAPLACE uses a rectangular grid of elements in the calculation of solutions to two dimensional boundary value problems. The program assigns place holders to elements between, above, or below subdomains to form a rectangular patten of elements. This procedure increases the number of elements actually held in memory. This increases in the number of elements stored in memory decreases the calculating capability of this program for any subdomain arrangements that do not combine form a rectangle.

Recommendations

This program has the ability to be expanded, and used to solve various types of second order boundary problems. Below are five recommendations regarding this program.

1. The limitation on mesh size should be overcome. Further study of Microsoft Basic 7.1 and Fortran 5.1 compilers might lead to a way to compile program LAPLACE so that extended memory could be taken advantage of.
2. The effectiveness of the program LAPLACE to deal with solutions to other second order, elliptic, two dimensional boundary value problems than represented by Laplace's equation needs to be validated.
3. The development of a graphics program internal to the program LAPLACE should be introduced. This would allow for an increase in the resolution of graphical outputs. If this program ran internal to LAPLACE, memory used by the GRAPHICS program in DOS 5.0 would be available to the program LAPLACE.
4. The time taken to generate the flow net needs to be decreased if the limitation of size is overcome. If meshes are calculated with the availability of extended memory, the time taken to interpolate all these nodes could increase dramatically, requiring faster computation for graphical output.
5. A subdomain entry procedure should be developed that would allow the program to only store elements required to calculate the solution to the two dimensional problem. By only storing elements in the domain of the problem solutions to larger problems could be calculated, and the computers memory could be more efficiently used.

APPENDICES

APPENDIX A

LAPLACE PROGRAM

Note: This program was developed by C. L. Arnold from a FORTRAN code by W. F. Caroll, as explained in chapter 3, pg 30 of this paper.

LAPLACE PROGRAM

```
==================Main Program================

'    PROGRAM LAPLACE.BAS BY C. L. Arnold, Civil Engineering, UCF
'
'    THIS FINITE ELEMENT CODE SOLVES A GENERAL 2nd ORDER,
'    LINEAR, ELLIPTC SCALAR PARTIAL DIFFERENTIAL EQUATION
'     OF THE FORM:
'
'     -DEL(DOT)(K(X,Y)*DEL(U))+C(X,Y)*(DU/DX+DU/DY)+
'          B(X,Y)*U(X,Y)=F(X,Y)
'
'     WHEN C,B, AND F=0 AND K = 1 THE EQUATION IS REPRESENTED
'     BY   LAPLACE'S EQUATION:   - DEL SQUARED (U) = 0
'
'     THE FUNCTIONS C, B, F, AND K  ARE CONTANED IN
'     SUBROUTINE COEF AND SET TO SOLVE LAPLACES EQUATION
'
'    THE CODE RUNS TOTALLY INTERACTIVELY;
'    THE USER PROVIDES DATA AND MAKES CHOICES ON DEMAND
'    (AS REQUESTED ON THE CRT DISPLAY)
'
'    THE DOMAIN OF X AND Y MUST BE SUBDIVIDED INTO 1-16
'    SUBDOMAINS; DATA IS INPUT BY SUBDOMAIN.
'
'    REGARDING ELEMENTS AND SUBDOMAINS:
'    - ONLY QUADRILATERAL ISOPARAMETRIC ELEMENTS  ARE
'      ALLOWED
'    - THESE MAY BE 4-NODE (LINEAR), 8-NODE
'      QUADRATIC/SERENDIPITY), OR 9-NODE
'      (BIQUADRATIC/LAGRANGE) ELEMENTS
'    - SUBDOMAINS MUST BE SIMILAR IN SHAPE TO THE ELEMS
'      THEY CONTAIN;
'    - THEIR KEY NODES MUST BE PLACED SIMILARLY TO THEIR
'      ELEM NODES
'    - THE KEY NODES ON THE COMMON BOUNDARY OF TWO ADJACENT
'      SUBDOMAINS MUST COINCIDE.
'    - ADJACENT SUBDOMAINS MUST HAVE THE SAME NR OF ROWS OR
'      COLS OF ELEMENTS AT THEIR COMMON BOUNDARIES SO
'      ELEMENT NODES AT COMMON SUBDOMAIN BOUNDARIES WILL
```

```
'     COINCIDE.
'   - THE LOWER LEFT SUBDOMAIN IS SUBDOMAIN NR 1.
'     NO  OTHER SUBDOMAIN MAY BE LOWER THAN SUBDOMAIN NR 1.
'   - THE POSITION OF THE FIRST ROW OF ELEMENTS IS
'     NUMBERED ONE AND THE ROWS ARE CONSECTIVLY NUMBERED
'     FROM THE LEFT
'
'   THE NR OF ELEMENTS ALLOWED DEPENDS ON THEIR TYPE (4 OR
'   8/9-NODE)  AND THEIR ARRANGEMENT (RECTANGULAR,
'   COLUMNAR, OR  IRREGULAR ARRAYS).
'   A FICTITIOUS RECTANGULAR ARRAY OF THE ELEMS AND THE
'   BLANK SPACES IN THE DOMAIN, WHERE THE LARGEST NR OF
'   ELEMS IN ANY COL IN THE ACTUAL ARRAY IS THE NR ROWS IN
'   THE FICTITIOUS ARRAY AND THE LARGEST NR OF ELEMS IN
'   ANY ROW OF THE ACTUAL ARRAY IS THE NR OF COLS IN THE
'   FICTITIOUS ARRAY, MUST NOT GENERATE MORE ARRAYS THAN
'   MEMEORY CAN HANDLE

'   THE MAX REQUIRED BAND-WIDTH (IBW) OF THE GLOBAL
'   STIFFNESS MATRIX, GK,IS DETERMINED BY THE ROW OF THE
'   ARRAY OF ELEMENTS IN THE DOMAIN WHICH POSSESSES THE
'   LARGEST NR OF ELEMENTS. FOR ANY ELEMENT IN THIS ROW,
'   IBW EQUALS TWICE THE GLOBAL NODE NR OF ELEMENT NODE 3
'   MINUS TWICE THE GLOBAL NODE NR OF ELEMENT NODE 1 PLUS
'   ONE. FOR EXAMPLE, FOR AN ELEMENT ARRAY WHOSE MAX ROW
'   HAS:
'      1 4-NODE ELEM,  IBW= 7
'     15 4-NODE ELEMS, IBW=35
'      1 8-NODE ELEM,  IBW=15
'      7 8-NODE ELEMS, IBW=51
'      1 9-NODE ELEM,  IBW=17
'      7 9-NODE ELEMS, IBW=65
'
'   DATA ON BC ARE ENTERED BY BOUNDARY STRIPS - THE SINGLE
'   ROWS OR COLS OF ELEMS ON THE PERIMETER OF THE DOMAIN;
'   1-25 STRIPS MAY BE USED.
'   THE BC DATA IN A STRIP (SAME FOR EACH ELEM IN THE
'   STRIP) ARE:
'    - KIND OF BC (TYPE 1:ESSENTIAL; TYPE 2:NATURAL; TYPE
'      3:MIXED)
'    - VALUE OF BC (U:TYPE 1; DU/DN:TYPE 2 OUTWARD IS
'      POSITIVE; H AND UE:TYPE 3 WHERE DU/DN = H*(U-UE); H
'      IS POSITIVE)
```

```
'     - ELEM SIDE NR ON WHICH BC ARE APPLIED (NR 1 FOR ELEM
'       NODES 1-2, NR 2 FOR 2-3, NR 3 FOR 3-4, AND NR 4 FOR 4-1)
'     - LOWEST ELEM NR IN THE STRIP
'     - HIGHEST ELEM NR IN THE STRIP
'     - INCREMENTAL VARIATION IN ELEM NR - GOING FROM
'       LOWEST TO HIGHEST.
'
'    BC ARE APPLIED AT ELEM NODES ON THE APPLICABLE ELEM
'    SIDE; ESSECTIAL  BC MAY BE AT BOTH CORNER NODES ON THE
'    SIDE OR JUST THE THE 1ST NODE (SEE APPLYBC); ELEMS
'    WITH BC ON 2 (OR 3) SIDES MUST APPEAR IN THE 2 (OR 3)
'    ADJOINING STRIPS OVERLAPPED ON THE ELEMENT.
'    NATURAL BC WITH A VALUE = 0 NEED NOT BE ENTERED.
'
'    INTEGRATION IS BY GUASSIAN QUADRATURE AT 4 OR 9 POINTS
'    IN EACH ELEM.SINCE THE TYPE OF ELEM MUST BE UNIFORM
'    THROUGHOUT THE DOMAIN, SO MUST BE THE ORDER OF
'    INTEGRATION.
'
'-------------------------------------------------------------------------------

REM $DYNAMIC

      DECLARE SUB ENDAT (NAME$)
      DECLARE SUB SIZE (NROW%, NCOL%)
      DECLARE SUB MESH (NROW%, NCOL%, NGLO%(), XYGLO#(),
      IGLO%)
      DECLARE SUB SHAPE (L%)
      DECLARE SUB BAND (NROW%, NCOL%, NGLO%(), XYGLO#(),
      IBW%,_
      IGLO%, MDOF%, MBW%)
      DECLARE SUB ELMDAT (IRE%, NROW%, NCOL%, NGLO%(),_
      XYGLO#())
      DECLARE SUB RBC ()
      DECLARE SUB BCDAT (NB%)
      DECLARE SUB ASSMB (NROW%, NCOL%, NGLO%(), XYGLO#(),_
      IGLO%, IBW%, GF#(), GK#())
      DECLARE SUB SETINT ()
      DECLARE SUB BCINT ()
      DECLARE SUB ELEM (IRE%)
      DECLARE SUB APLYBC (IRE%)
      DECLARE SUB BSHAPE (INTB%)
      DECLARE SUB SOLVE (IGLO%, IBW%, GF#(), GK#(), U#())
```

```
     DECLARE SUB PROUT (NROW%, NCOL%, NGLO%(), XYGLO#(),_
     IGLO%, U#())
     DECLARE SUB COEF (GX#, GY#)
     DECLARE SUB PLOT2D (NROW%, NCOL%, NGLO%(), XYGLO#(),_
     IGLO%, PAUSE$)
     DECLARE SUB CONTOUR (NROW%, NCOL%, NGLO%(), XYGLO#(),_
     IGLO%, U#())
     DECLARE SUB EXIST ()
     DECLARE SUB ZSORT ()
     DECLARE SUB FLOW1 (NROW%, NCOL%, NGLO%(), XYGLO#(),_
     IGLO%, U#())

     COMMON /CELM/ NSD%, NRSD%, NCSD%(), KELM%, IELM%,_
     ICR%(), XSD#()
     COMMON /CESH/ XXI#, EETA#
     COMMON /CHAPE/ PSI#(), DPSI#()
     COMMON /SIZEA/ NCOLE%(), NRE%
     COMMON /CINTA/ XIETA#(), WT#()
     COMMON /CBC/ NRB%(), KBC%(), VBC#(), ISIDE%(), NRBE%()
     COMMON /KOEF/ F#, AK#, C#, B#
     COMMON /EFK/ EF#(), EK#()
     COMMON /CEDAT/ NELM%(), XYELM#()
     COMMON /BHAPE/ BPSI#(), BDPSI#()
     COMMON /BINT/ BXI#(), BWT#()
     COMMON /SHA/ VCT%(), EDFL$

     CLS
10   MDOF% = 1500
     IBW% = 500
     PRINT : PRINT : PRINT : PRINT
20   PRINT SPC(11); "Program LaPlace - by C. L. Arnold,  UCF Civil _
     Engineering"
     PRINT : PRINT
     PRINT SPC(15); "Finite Element Solution of LaPlaces  Equation"
     PRINT
     PRINT SPC(26); "del squared (U) = 0"
     PRINT
30   PRINT SPC(8); "Max nr of (real + blank) nodes is ";_
     MDOF%; " Max bandwidth is "; MBW%
     PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT
     INPUT "hit enter to continue"; PAUSE$

40   CALL ENDAT(NAME$)
```

```
        CALL SIZE(NROW%, NCOL%)
        DIM NGLO%(NROW%, NCOL%), XYGLO#(2, NROW%, NCOL%)

        CALL MESH(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%)
        CALL PLOT2D(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%,_
        PAUSE$)
            IF PAUSE$ = "n" OR PAUSE$ = "N" THEN GOTO 60
        CALL RBC
        CALL BAND(NROW%, NCOL%, NGLO%(), XYGLO#(), IBW%, IGLO%,_
        MDOF%, MBW%)
            DIM GF#(IGLO%), GK#(IGLO%, IBW%)
            DIM BPSI(3), BDPSI(3)

        CALL ASSMB(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%, IBW%,_
        GF#( ), GK#( ))
            DIM U#(IGLO%)

        CALL SOLVE(IGLO%, IBW%, GF#( ), GK#( ), U#( ))
        CALL PROUT(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%, U#())
        CALL CONTOUR(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%,_
        U#())
        CALL ZSORT
        CALL FLOW1(NROW%, NCOL%, NGLO%(), XYGLO#(), IGLO%, U#())

        SCREEN 0
        CLS
50      PRINT : PRINT
        PRINT SPC(1); "The solution for the mesh is in file "; NAME$ + ".msh"
        PRINT
        PRINT SPC(1); "The contour data is in file "; NAME$ + ".ctr"
        PRINT
        PRINT SPC(1); "The flow data is in file "; NAME$ + ".flo"
        PRINT
        PRINT SPC(1); "The mesh data is in file"; NAME$ + ".dat"
60      PRINT : PRINT
        PRINT "You will have to restart program to reset arrays"
70

        END

        REM $STATIC
```

```
----------Subbroutines----------

'================SUB APLYBC==============

SUB APLYBC (IRE%)

     SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
     SHARED EF#( ), EK#( )
     SHARED NELM%( ), XYELM#( )
     SHARED NRB%, KBC%( ), VBC#( ), ISIDE%( ), NRBE%( )
     SHARED BPSI#( ), BDPSI#( )
     SHARED BXI#( ), BWT#( )

     DIM IRB%(3), ISZ%(3), XYM#(2, 3)

'-----IDENTIFY BOUNDARY STRIPS IN WHICH ELEMENT IS LOCATED-----

     IST% = 0
     FOR IB% = 1 TO NRB%
          FOR NEL% = NRBE%(1, IB%) TO NRBE%(2, IB%) STEP_
               NRBE%(3, IB%)
               IF NEL% <> IRE% GOTO 100
               IST% = IST% + 1
               IRB%(IST%) = IB%
100       NEXT NEL%
     NEXT IB%

     IF IST% = 0 THEN EXIT SUB

'-----LOOP ON APPLICABLE ELEMENT SIDES------

     FOR ISE% = 1 TO IST%
          IB% = IRB%(ISE%)
          ISZ%(1) = ISIDE%(IB%)
          ISZ%(2) = ISZ%(1) + 1
          ISZ%(3) = ISZ%(1) + 4
          IF ISZ%(1) = 4 THEN ISZ%(2) = 1
          IF KBC%(IB%) <> 1 THEN GOTO 120


'-----APPLY ESSENTIAL BC TO BOTH CORNER NODES ON THE SIDE-----
'    (DO 110 I=1,3)  OR TO JUST THE 1ST CORNER NODE ON THE SIDE
```

```
'    (DO 110 I=1,3,2); MID-SIDE NODE IS TAKEN CARE OF
'    AUTOMATICALLY AS APPLICABLE

         FOR I% = 1 TO 3
              IE% = ISZ%(I%)
              IF KELM% = 4 AND I% = 3 GOTO 110
              EK#(IE%, IE%) = EK#(IE%, IE%) + (1D+15)
              EF#(IE%) = EF#(IE%) + VBC#(1, IB%) * (1D+15)
110      NEXT I%

         GOTO 160

'-----APPLY NATURAL OR MIXED BC-----

         120  VB# = VBC#(1, IB%)
         IF KBC%(IB%) = 3 THEN VB# = -VB# * VBC#(2, IB%)

'-----OBTAIN X-Y COORDS AT THE 3 NODES ON THE ELEM SIDE-----

         FOR I% = 1 TO 3
              INTA% = ISZ%(I%)
              FOR J% = 1 TO 2
                   XYM#(J%, I%) = XYELM#(J%, INTA%)
              NEXT J%
         NEXT I%

         IF KELM% <> 4 GOTO 130

         FOR J% = 1 TO 2
              XYM#(J%, 3) = (XYM#(J%, 1) + XYM#(J%, 2)) / 2#
         NEXT J%


'-----CALCULATE THE JACOBIAN FOR INTEGRATION ALONG THE-----
'    ELEM SIDE

130      FOR INTB% = 1 TO 3
              CALL BSHAPE(INTB%)
              DXDS# = XYM#(1, 1) * BDPSI#(1) + XYM#(1, 2) * BDPSI#(2)_
              + XYM#(1, 3) * BDPSI#(3)
              DYDS# = XYM#(2, 1) * BDPSI#(1) + XYM#(2, 2) * BDPSI#(2)_
              + XYM#(2, 3) * BDPSI#(3)
              DJAC# = SQR(DXDS# ^ 2 + DYDS# ^ 2)
```

```
            IR% = 3
            IF KELM% <> 4 GOTO 140


'-----CONVERT 1D QUADRATIC SHAPE FUNCTIONS TO 1D LINEAR-----
'    SHAPE FUNCTIONS FOR 4-NODE LINEAR ELEMS

            IR% = 2
            FOR I% = 1 TO 2
                BPSI#(I%) = BPSI#(I%) + .5# * BPSI#(3)
            NEXT I%

'-----INTEGRATE ALONG THE ELEM BOUNDARY-----

140         FOR I% = 1 TO IR%
                IE% = ISZ%(I%)
                EF#(IE%) = EF#(IE%) - VB# * BPSI#(I%) * BWT#(INTB%)_
                * DJAC#
                IF KBC%(IB%) = 2 GOTO 150
                FOR J% = 1 TO IR%
                    JE% = ISZ%(J%)
                    EK#(IE%, JE%) = EK#(IE%, JE%) + VBC#(1, IB%) *_
                    BPSI#(I%) * BPSI#(J%) * BWT#(INTB%) * DJAC#
                NEXT J%
150         NEXT I%
        NEXT INTB%
160 NEXT ISE%

    END SUB

'================SUB ASSMB================

    SUB ASSMB (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IGLO%,_
    IBW%,GF#( ), GK#( ))

    SHARED NCOLE%( ), NRE%
    SHARED NELM%( ), XYELM#( )
    SHARED EF#( ), EK#( )
    SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )

    DIM EF#(9), EK#(9, 9)

'-----INITIALIZE-----
```

```
        FOR I = 1 TO IGLO%
            GF#(I) = 0#
            FOR J = 1 TO IBW%
                GK#(I, J) = 0#
            NEXT J
        NEXT I

        CALL SETINT
        CALL BCINT

'-----LOOP ON THE ELEMENTS-----

        FOR IRE% = 1 TO NRE%
            CALL ELMDAT(IRE%, NROW%, NCOL%, NGLO%(), XYGLO#())
            CALL ELEM(IRE%)
            CALL APLYBC(IRE%)

            LOCATE 10, 26: PRINT "*****************************"
            LOCATE 11, 26: PRINT "*  BUILDING GLOBAL MATRIX  *"
            LOCATE 12, 26: PRINT "*         "; TIME$; "        *"
            LOCATE 13, 26: PRINT "*****************************"

'------ASSEMBLE THE GLOBAL FORCE VECTOR-----

            FOR I = 1 TO KELM%
                IG% = NELM%(I)
                GF#(IG%) = GF#(IG%) + EF#(I)

'-----ASSEMBLE THE GLOBAL STIFFNESS MATRIX-----

                FOR J = 1 TO KELM%
                    JG% = NELM%(J) - IG% + (IBW% / 2 - .5) + 1
                    IF JG% < 0 OR JG% > IBW% GOTO 200
                    GK#(IG%, JG%) = GK#(IG%, JG%) + EK#(I, J)
200             NEXT J
            NEXT I
        NEXT IRE%

        END SUB

'==============SUB BAND====================
```

```
SUB BAND (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IBW%, IGLO%,_
MDOF%, MBW%)

SHARED NCOLE%( ), NRE%
SHARED NELM%( ), XYELM#( )
SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )

DIM NELM%(9), XYELM#(2, 9)

'-----CALCULATE THE MAX REQUIRED BAND WIDTH (IBW%) FOR THE -----
'    GLOBAL STIFFNESS MATRIX, GK#(IGLO%,IBW%)

CLS
IBW% = 1
FOR IRE% = 1 TO NRE%

    CALL ELMDAT(IRE%, NROW%, NCOL%, NGLO%(), XYGLO#())

    FOR I = 1 TO KELM%
        ITBW% = 2 * (NELM%(3) - NELM%(1)) + 1
        IF IBW% < ITBW% THEN IBW% = ITBW%
    NEXT I

    LOCATE 10, 26: PRINT "*****************************"
    LOCATE 11, 26: PRINT "*     BANDING MATRIX      *"
    LOCATE 12, 26: PRINT "*        "; TIME$; "       *"
    LOCATE 13, 26: PRINT "*****************************"

NEXT IRE%

CLS
NX% = NROW% * NCOL%
IF IBW% > MBW% OR NX% > MDOF% THEN 300 ELSE 310

300 PRINT " You have exceeded the limits for max bandwidth or"
    PRINT " (real + blank) nodes": PRINT
    INPUT " Want to re-enter domain and subdomain data or just stop? r/s"; IY$
    PRINT
    IF IY$ = "S" OR IY$ = "s" THEN STOP
    CALL ENDAT(EDFL$)

310 END SUB
```

```
'==============SUB BCDAT====================

    SUB BCDAT (NB%)

    SHARED NRB%, KBC%( ), VBC#( ), ISIDE%( ), NRBE%( )

    REDIM NBC%(3): NBC%(1) = 1: NBC%(2) = 1: NBC%(3) = 2

'-----READ SIX BC DATA FOR TYPE 1 OR 2 STRIP, SEVEN BC DATA FOR-----
'     TYPE 3 STRIP

    NS% = 1
    NBS% = NRB%
    IF NB% <> 0 THEN NS% = NB%
    IF NB% <> 0 THEN NBS% = NB%
    FOR J = NS% TO NBS%
        PRINT : PRINT " Boundary Strip,"; J

        INPUT KB%
        FOR K = 1 TO NBC%(KB%)
            INPUT VBC#(K, J)
        NEXT K

        INPUT ISIDE%(J)
        FOR I = 1 TO 3
            INPUT NRBE%(I, J)
        NEXT I

        KBC%(J) = KB%
    NEXT J

    END SUB

'==============SUB BCINT====================

    SUB BCINT

    SHARED BXI#( ), BWT#( )
    DIM BXI#(3), BWT#(3)

    BXI#(1) = -(SQR(3# / 5#))
    BXI#(2) = -BXI#(1)
    BXI#(3) = 0#
```

```
        BWT#(1) = 5# / 9#
        BWT#(2) = BWT#(1)
        BWT#(3) = 8# / 9#

        END SUB

'=============SUB BSHAPE=====================

        SUB BSHAPE (INTB%)

        SHARED BXI#( ), BWT#( )
        SHARED BPSI#( ), BDPSI#( )

'-----EVALUATE 1D QUADRATIC SHAPE FUNCTIONS FOR ELEM SIDES-----

        BPSI#(1) = BXI#(INTB%) * (BXI#(INTB%) - 1#) * .5#
        BPSI#(2) = BXI#(INTB%) * (BXI#(INTB%) + 1#) * .5#
        BPSI#(3) = 1# - BXI#(INTB%) ^ 2#
        BDPSI#(1) = BXI#(INTB%) - .5#
        BDPSI#(2) = BXI#(INTB%) + .5#
        BDPSI#(3) = -2# * BXI#(INTB%)

        END SUB

'================SUB COEF====================

        SUB COEF (GX#, GY#)

        SHARED F#, AK#, C#, B#

'-----THE LOAD FUNCTION F AND COEFFICIENTS AK, AND B MUST BE-----
'       EXPRESSED AS FUNCTIONS OF THE GLOBAL X-Y COORDS, GX AND
'       GY. IT MUST BE POSSIBLE TO CALCULATE THEIR VALUES AT EACH
'       INTEGRATION POINT WITHIN EACH ELEMENT.

        F# = 0#
        AK# = 1#
        C# = 0#
        B# = 0#

        END SUB

'================SUB CONTOUR=================
```

```
      SUB CONTOUR (NROW%, NCOL%, NGLO%( ), XYGLO#( ),_
      IGLO%, U#())

      REDIM XX#(0 TO NROW% + 1, O TO NCOL% + 1, 2)
      REDIM NNGLO%(0 TO NROW% + 1, 0 TO NCOL% + 1)
      DIM FF#(NROW% + 1, NCOL% + 1)
      DIM XX(IGLO%), YY(IGLO%)

'-----CHANGE ASPECT RATIO  IF NOT A SQUARE THEN SQUARE-----

      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              NNGLO%(I, J) = NGLO%(I, J)
              FOR M = 1 TO 2
                  XX#(I, J, M) = XYGLO#(M, I, J)
              NEXT M
              IF NGLO%(I, J) = 0 THEN 400
              IJ = IJ + 1
              FF#(I, J) = U#(IJ)
400       NEXT J
      NEXT I

'-----SET SCREEN-----

410   SCREEN 0
      CLS
      KEY OFF: DEFINT I-N
      SCREEN 11: F$ = "####.##"
      ASP = .46
      CLS

      XMAX = XX#(1, 1, 1)
      YMAX = XX#(1, 1, 2)
      XMIN = XX#(1, 1, 1)
      YMIN = XX#(1, 1, 2)
      FMAX = FF#(1, 1)
      FMIN = FF#(1, 1)

      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              IF XMAX < XX#(I, J, 1) THEN XMAX = XX#(I, J, 1)
              IF YMAX < XX#(I, J, 2) THEN YMAX = XX#(I, J, 2)
```

```
            IF XMIN > XX#(I, J, 1) THEN XMIN = XX#(I, J, 1)
            IF YMIN > XX#(I, J, 2) THEN YMIN = XX#(I, J, 2)
            IF NNGLO%(I, J) = 0 THEN 420
            IF FMAX < FF#(I, J) THEN FMAX = FF#(I, J)
            IF FMIN > FF#(I, J) THEN FMIN = FF#(I, J)
420       NEXT J
    NEXT I
    NN = 1

    LOCATE 16, 25: INPUT "Number of Contour Lines"; NCL
    LOCATE 18, 25: INPUT "Overlap mesh on Contour Lines (Y/N)"; msh$

    STP = (FMAX - FMIN) / (NCL)
    CLS

    XL = (XMAX - XMIN)
    YL = (YMAX - YMIN)
    X0 = XMIN - XL / 10
    Y0 = YMIN - YL / 10

    VIEW (51, 1)-(639, 430)

    AA = 538 * ASP / 167

    IF XL / YL > AA THEN YL = XL / AA
    IF XL / YL < AA THEN XL = YL * AA
    XMAX = X0 + 1.2 * XL: YMAX = Y0 + 1.2 * YL
    WINDOW (X0, Y0)-(XMAX, YMAX)
    CLS
    IF msh$ = "Y" OR msh$ = "y" THEN GOTO 470

'-----FIND BOUNDRY LINES-----

    FOR I = 1 TO (NROW%)
        FOR J = 1 TO (NCOL%)

            IF NNGLO%(I, J) = 0 THEN 460
            X1 = XX#(I, J, 1)
            Y1 = XX#(I, J, 2)

            IF NNGLO%(I + 1, J + 1) = 0 THEN 430
            IF NNGLO%(I + 1, J) = 0 OR NNGLO%(I - 1, J) = 0 THEN 430
            IF NNGLO%(I - 1, J + 1) = 0 AND NNGLO%(I - 1, J) < > 0_
```

```
            THEN 430 ELSE 440

430         X2 = XX#(I, J + 1, 1)
            Y2 = XX#(I, J + 1, 2)
            IF X2 = 0 THEN 440
            LINE (X1, Y1)-(X2, Y2)

440         IF NNGLO%(I + 1, J + 1) = 0 THEN 450
            IF NNGLO%(I, J + 1) = 0 OR NNGLO%(I, J - 1) = 0 THEN 450
            IF NNGLO%(I + 1, J - 1) = 0 AND NNGLO%(I, J - 1) < > 0_
            THEN 450 ELSE 460

450         X2 = XX#(I + 1, J, 1)
            Y2 = XX#(I + 1, J, 2)
            IF Y2 = 0 THEN 460
            LINE (X1, Y1)-(X2, Y2)
460       NEXT J
    NEXT I

    GOTO 500

'-----DRAW MESH-----

470 FOR I = 1 TO (NROW%)
        FOR J = 1 TO (NCOL%)

            IF NNGLO%(I, J) = 0 THEN 490
            X1 = XX#(I, J, 1)
            Y1 = XX#(I, J, 2)

            IF NNGLO%(I, J + 1) = 0 THEN 480
            IF J = NCOL% THEN 480
            X2 = XX#(I, J + 1, 1)
            Y2 = XX#(I, J + 1, 2)

            LINE (X1, Y1)-(X2, Y2)

480         IF I = NROW% THEN 490
            IF NNGLO%(I + 1, J) = 0 THEN 490
            X2 = XX#(I + 1, J, 1)
            Y2 = XX#(I + 1, J, 2)

            LINE (X1, Y1)-(X2, Y2)
```

```
490       NEXT J
      NEXT I


'-----CONTOUR PLOTTING-----

500   OPEN "CONTR.TMP" FOR OUTPUT AS #4

      PRINT #4, X0, Y0
      PRINT #4, XMAX, YMAX
      PRINT #4, FMIN
      BCOUNT = 1
      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              FF = FF#(I, J)
              IF FMIN = FF THEN GOTO 501 ELSE 502
501           BCXX# = XX#(I, J, 1): BCYY# = XX#(I, J, 2)
              IF BCOUNT <> 1 THEN GOTO 502
              PRINT #4, BCXX#, BCYY#
              BCOUNT = BCOUNT + 1
502       NEXT J
      NEXT I
      PRINT #4, BCXX#, BCYY#
      PRINT #4, 0, 0

      REDIM XXX(2), YYY(2)
      FOR CTL = (FMIN + STP) TO FMAX STEP STP
          PRINT #4, CTL
          FOR I = 1 TO (NROW% - 1)
              FOR J = 1 TO (NCOL% - 1)

                  N = 0
                  IF NNGLO%(I + 1, J) = 0 THEN 590
                  IF NNGLO%(I, J + 1) = 0 THEN 590
                  IF NNGLO%(I + 1, J + 1) = 0 THEN 590
                  IF FF#(I, J) = 0 THEN 590

                  IF FF#(I, J) < CTL AND FF#(I, J + 1) > CTL THEN 510
                  IF FF#(I, J) > CTL AND FF#(I, J + 1) < CTL THEN 510_
                  ELSE 520
510               N = N + 1
                  X1 = XX#(I, J, 1)
                  Y1 = XX#(I, J, 2)
```

```
            X2 = XX#(I, J + 1, 1)
            Y2 = XX#(I, J + 1, 2)
            PP = (CTL - FF#(I, J)) / (FF#(I, J + 1) - FF#(I, J))

            XXX(N) = X1 + (X2 - X1) * PP
            YYY(N) = Y1 + (Y2 - Y1) * PP

520         IF FF#(I, J) < CTL AND FF#(I + 1, J) > CTL THEN 530
            IF FF#(I, J) > CTL AND FF#(I + 1, J) < CTL THEN 530_
            ELSE 540
530         N = N + 1
            X1 = XX#(I, J, 1)
            Y1 = XX#(I, J, 2)
            X2 = XX#(I + 1, J, 1)
            Y2 = XX#(I + 1, J, 2)
            PP = (CTL - FF#(I, J)) / (FF#(I + 1, J) - FF#(I, J))

            XXX(N) = X1 + (X2 - X1) * PP
            YYY(N) = Y1 + (Y2 - Y1) * PP

540         IF FF#(I, J + 1) < CTL AND FF#(I + 1, J + 1) > CTL_
            THEN 550
            IF FF#(I, J + 1) > CTL AND FF#(I + 1, J + 1) < CTL_
            THEN 550 ELSE 560

550         N = N + 1
            X1 = XX#(I, J + 1, 1)
            Y1 = XX#(I, J + 1, 2)
            X2 = XX#(I + 1, J + 1, 1)
            Y2 = XX#(I + 1, J + 1, 2)
            PP = (CTL - FF#(I, J + 1)) / (FF#(I + 1, J + 1) - FF#(I,_
            J+1))

            XXX(N) = X1 + (X2 - X1) * PP
            YYY(N) = Y1 + (Y2 - Y1) * PP

560         IF FF#(I + 1, J) < CTL AND FF#(I + 1, J + 1) > CTL_
            THEN 570
            IF FF#(I + 1, J) > CTL AND FF#(I + 1, J + 1) < CTL_
            THEN 570 ELSE 580
570         N = N + 1
            X1 = XX#(I + 1, J, 1)
            Y1 = XX#(I + 1, J, 2)
```

```
                X2 = XX#(I + 1, J + 1, 1)
                Y2 = XX#(I + 1, J + 1, 2)
                PP = (CTL - FF#(I + 1, J)) / (FF#(I + 1, J + 1) - FF#(I +_
                1, J))


                XXX(N) = X1 + (X2 - X1) * PP
                YYY(N) = Y1 + (Y2 - Y1) * PP

580             IF XXX(1) = XXX(2) AND YYY(1) = YYY(2) THEN 590
                LINE (XXX(2), YYY(2))-(XXX(1), YYY(1))

                PRINT #4, XXX(1), YYY(1)
                PRINT #4, XXX(2), YYY(2)
                XXX(2) = XXX(1)
                YYY(2) = YYY(1)

590             NEXT J
            NEXT I
            PRINT #4, 0, 0
        NEXT CTL
        PRINT #4, -100000

        PRINT #4, FMAX
        BCOUNT = 1
        FOR I = 1 TO NROW%
            FOR J = 1 TO NCOL%
                FF = FF#(I, J)
                IF FMAX = FF THEN GOTO 591 ELSE 592
591             BCXX# = XX#(I, J, 1): BCYY# = XX#(I, J, 2)
                IF BCOUNT < > 1 THEN GOTO 592
                PRINT #4, BCXX#, BCYY#
                BCOUNT = BCOUNT + 1
592         NEXT J
        NEXT I
        PRINT #4, BCXX#, BCYY#
        CLOSE #4

        LOCATE 27, 5:
        PRINT "BOUNRY CONDITIONS, MIN = "; FMIN; ",MAX = "; FMAX
        LOCATE 28, 5:
        PRINT "CONTOUR INTERVAL = "; : PRINT USING "###.####"; STP
        LOCATE 29, 5:
```

```
      INPUT "WANT TO REDRAW WITH NEW CONTOUR INTERVAL _
      (Y/N)"; RED$
      IF RED$ = "Y" OR RED$ = "y" THEN GOTO 410

      CLS
      SCREEN 0
      CLS

      END SUB

    DEFSNG I-N
'=====================SUB ELEM================

SUB ELEM (IRE%)

      SHARED NELM%( ), XYELM#( )
      SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
      SHARED PSI#( ), DPSI#( )
      SHARED XIETA#( ), WT#( )
      SHARED F#, AK#, C#, B#
      SHARED EF#( ), EK#( )

      REDIM DXDS#(2, 2), DSDX#(2, 2), DPSIX#(KELM%), DPSIY#(KELM%)


'-----INITIALIZE-----

      FOR I = 1 TO 9
            EF#(I) = 0#
            FOR J = 1 TO 9
                  EK#(I, J) = 0#
            NEXT J
      NEXT I

'-----LOOP ON THE INTEGRATION POINTS-----

      FOR L% = 1 TO IELM%
            CALL SHAPE(L%)
            GX# = 0#
            GY# = 0#
            FOR I = 1 TO KELM%
                  GX# = GX# + XYELM#(1, I) * PSI#(I)
                  GY# = GY# + XYELM#(2, I) * PSI#(I)
```

```
        NEXT I

        CALL COEF(GX#, GY#)

'-----CALCULATE DXDS#-----

        FOR I = 1 TO 2
            FOR J = 1 TO 2
                DXDS#(I, J) = 0#
                FOR K = 1 TO KELM%
                    DXDS#(I, J) = DXDS#(I, J) + XYELM#(I, K) *_
                    DPSI#(J, K)
                NEXT K
            NEXT J
        NEXT I

'-----CALCULATE DSDX#-----

        DETJ# = DXDS#(1, 1) * DXDS#(2, 2) - DXDS#(1, 2) * DXDS#(2, 1)

        IF DETJ# < 0 GOTO 600

        DSDX#(1, 1) = DXDS#(2, 2) / DETJ#
        DSDX#(1, 2) = -DXDS#(1, 2) / DETJ#
        DSDX#(2, 1) = -DXDS#(2, 1) / DETJ#
        DSDX#(2, 2) = DXDS#(1, 1) / DETJ#

'-----CALCULATE D(PSI#)/DX AND D(PSI#)/DY-----

        FOR I = 1 TO KELM%
            DPSIX#(I) = DPSI#(1, I) * DSDX#(1, 1) + DPSI#(2, I) *_
            DSDX#(2, 1)
            DPSIY#(I) = DPSI#(1, I) * DSDX#(1, 2) + DPSI#(2, I) *_
            DSDX#(2, 2)
        NEXT I

'-----ACCUMULATE INTEGRATION POINT VALUES OF INTEGRALS-----
'    FOR  EK, THE ELEMENT STIFFNESS MATRIX

        FAC# = DETJ# * WT#(L%)

        FOR I = 1 TO KELM%
            EF#(I) = EF#(I) + F# * PSI#(I) * FAC#
```

```
            FOR J = 1 TO KELM%
                EK#(I, J) = EK#(I, J) + FAC# * (AK# * (DPSIX#(I) *_
                DPSIX#(J) + DPSIY#(I) * DPSIY#(J)) + C# * PSI#(I) *_
                (DPSIX#(J) + DPSIY#(J)) + B# * PSI#(I) * PSI#(J))
            NEXT J
        NEXT I
    NEXT L%

    EXIT SUB

600 PRINT : PRINT " ELEM NR,"; IRE%; ",  JACOBIAN=,"; DETJ#; ",  IT'S_
    BAD"

    INPUT "HIT RETURN TO CONTINUE"; PAUSE$
    STOP

    END SUB

'================SUB ELMDAT=================
    SUB ELMDAT (IRE%, NROW%, NCOL%, NGLO%( ), XYGLO#( ))

    SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
    SHARED NELM%( ), XYELM#( )

'-----IDENTIFY ROW AND COL INDICES OF ELEM LOWER L-H NODE-----

    NLQ% = 1
    IF KELM% <> 4 THEN NLQ% = 2
    N% = 0
    MROW% = NROW% - 1
    MCOL% = NCOL% - 1
    FOR I = 1 TO MROW% STEP NLQ%
        FOR J = 1 TO MCOL% STEP NLQ%
            IF NGLO%(I, J) = 0 GOTO 700
            IF NGLO%(I, (J + NLQ%)) = 0 GOTO 700
            IF NGLO%((I + NLQ%), J) = 0 GOTO 700
            IF NGLO%((I + NLQ%), (J + NLQ%)) = 0 GOTO 700
            N% = N% + 1
            IF N% < IRE% GOTO 700
            IE% = I
            JE% = J
            GOTO 710
700     NEXT J
```

```
    NEXT I

'-----PUT GLOBAL NODE NRS AND X-Y COORDS INTO ELEM ARRAYS-----

710 K = 1
    IE1% = IE% + 1
    JE1% = JE% + 1
    FOR I = 1 TO 2
        FOR J = 1 TO 9
            XYELM#(I, J) = 0#
        NEXT J
    NEXT I
    IF KELM% <> 4 GOTO 720

'-----LINEAR (4-NODE) ELEM-----

    FOR I = IE% TO IE1%
        FOR J = JE% TO JE1%
            IF I = IE% AND J = JE1% THEN K = 2
            IF I = IE1% AND J = JE1% THEN K = 3
            IF I = IE1% AND J = JE% THEN K = 4
            NELM%(K) = NGLO%(I, J)
            FOR L = 1 TO 2
                XYELM#(L, K) = XYGLO#(L, I, J)
            NEXT L
        NEXT J
    NEXT I

    EXIT SUB

'-----QUADRATIC/BIQUADRATIC (8/9-NODE) ELEMS-----

720 IE2% = IE% + 2
    JE2% = JE% + 2
    FOR I = IE% TO IE2%
        FOR J = JE% TO JE2%
            IF I = IE% AND J = JE2% THEN K = 2
            IF I = IE2% AND J = JE2% THEN K = 3
            IF I = IE2% AND J = JE% THEN K = 4
            IF I = IE% AND J = JE1% THEN K = 5
            IF I = IE1% AND J = JE2% THEN K = 6
            IF I = IE2% AND J = JE1% THEN K = 7
            IF I = IE1% AND J = JE% THEN K = 8
```

```
            IF I = IE1% AND J = JE1% THEN K = 9
            NELM%(K) = NGLO%(I, J)
            FOR L = 1 TO 2
                XYELM#(L, K) = XYGLO#(L, I, J)
            NEXT L
        NEXT J
    NEXT I

END SUB

'==========================SUB  ENDAT=================

    SUB ENDAT (NAME$)

    SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
    SHARED VCT%( ), EDFL$
    LET NAME$ = EDFL$
    CLS : PRINT : PRINT : PRINT : PRINT

    INPUT "Do you want to work with an existing data file (Y/N)"; Y$
    IF Y$ = "y" OR Y$ = "Y" THEN CALL EXIST ELSE 800
    GOTO 850

'-----ENTER MESH DATA BY SUBDOMAIN-----

800  INPUT "Name of new datafile"; EDFL$

    OPEN EDFL$ + ".MSH" FOR OUTPUT AS #1
    OPEN EDFL$ + ".TMP" FOR OUTPUT AS #2
    OPEN EDFL$ + ".DAT" FOR OUTPUT AS #3

    CLS : PRINT : PRINT
    INPUT " Enter the nr of rows of subdomains"; NRSD%

    REDIM NCSD%(NRSD%)
    REDIM VCT%(NRSD%)

    PRINT : PRINT
    PRINT "Enter the nr of subdomains in each row - start with row 1"
    PRINT "and the position of the lower left node of the first "
    PRINT "active subdomain in the cooresponding row"

    PRINT
```

```
      FOR I = 1 TO NRSD%
          PRINT "Subdomains in row #"; I;
          INPUT NCSD%(I)
          PRINT "Position of first active subdomain in row #"; I;
          INPUT VCT%(I)
          PRINT
      NEXT I
      NSD% = 0

      FOR I = 1 TO NRSD%
          NSD% = NSD% + NCSD%(I)
      NEXT I

      IF NSD% < 16 GOTO 810
      PRINT : PRINT
      PRINT " Nr of subdomains is "; NSD%; " the limit is 16"
      PRINT : PRINT
      GOTO 800
      PRINT : PRINT

      810  PRINT
      PRINT " For the domain, enter:": PRINT
      PRINT SPC(4); "- nr of nodes per elem (4,8 or 9)"
      PRINT SPC(4); "- nr of integration points per elem": PRINT
      INPUT KELM%: INPUT IELM%
      PRINT : PRINT
      INPUT "Are all domain entries correct? y/n"; IY$
      IF IY$ = "N" OR IY$ = "n" GOTO 800

      REDIM ICR%(2, NSD%), XSD#(2, KELM%, NSD%)

      PRINT : PRINT
      PRINT " Number subdomains lEFt to right - start with row 1"
      FOR K = 1 TO NSD%
820       CLS
          PRINT : PRINT : PRINT
          PRINT " For subdomain "; K; " enter:": PRINT
          PRINT SPC(4); " - nr of rows of elems in the subdomain"
          PRINT SPC(4); " - nr of elems per row in the subdomain": PRINT
          INPUT ICR%(2, K): INPUT ICR%(1, K)
          PRINT : PRINT
          PRINT " Enter x-y coords of key nodes of subdomain"; K
          PRINT SPC(4); "- 4,8,or 9 sets of x-y coords (linear,quadratic,or biquad)"
```

```
        PRINT SPC(6); "  do corners 1st, CCW from lower LH corner, then"
        PRINT SPC(6); "  do mid-side nodes, if any, CCW from lower mid-side_
        node"
        PRINT SPC(6); "  do center node last, if any.": PRINT
        FOR J = 1 TO KELM%
            FOR I = 1 TO 2
                INPUT XSD#(I, J, K)
            NEXT I
        NEXT J

        PRINT : PRINT
        PRINT "Are data for subdomain "; K; " correct? y/n": PRINT
        INPUT IY$
        IF IY$ = "N" OR IY$ = "n" GOTO 820
    NEXT K

    INPUT "Want to see your data entries? y/n"; IY$
    IF IY$ = "N" OR IY$ = "n" GOTO 840

'-----DISPLAY DATA ENTRIES AND PERMIT RE-ENTRY, IF DESIRED-----

    PRINT
    PRINT SPC(1); "Nr of subdomains is "; NSD%; "  nr of rows of subdomains_
    is"; NRSD%
    FOR K = 1 TO NSD%
        CLS
        PRINT : PRINT : PRINT : PRINT
        PRINT SPC(1); "Sub-"; SPC(6); "Elem"; SPC(6); "Elem"; SPC(6);_
        "Rows";
        SPC(6); "Cols"
        PRINT SPC(1); "Domain"; SPC(4); "Nodes"; SPC(4); "Int Pt"; SPC(5);
        "Elems"; SPC(5); "Elems"
        PRINT SPC(2); K; SPC(7); KELM%; SPC(7); IELM%; SPC(7);_
        ICR%(2, K); SPC(7); ICR%(1, K)
        PRINT
        PRINT SPC(10); "Key nodes of subdomain nr"; K
        PRINT
        PRINT "Coords";

        FOR I = 1 TO KELM%
            PRINT SPC(8); USING "###"; I;
        NEXT I
```

```
        PRINT
        PRINT ; "  X     ";
        FOR I = 1 TO KELM%
            PRINT SPC(5); USING "###.##"; XSD#(1, I, K);
        NEXT I

        PRINT
        PRINT ; "  Y     ";
        FOR I = 1 TO KELM%
            PRINT SPC(5); USING "###.##"; XSD#(2, I, K);
        NEXT I

        PRINT
        INPUT "Are these correct? y/n"; IY$: PRINT
        IF IY$ = "Y" OR IY$ = "y" GOTO 830

        PRINT
        PRINT "For subdomain"; K; " re-enter:"
        PRINT SPC(4); "- nr of rows of elems"
        PRINT SPC(4); "- nr of elems per row"
        PRINT SPC(4); "- x-y coords of its key nodes": PRINT
        FOR J = 1 TO KELM%
            INPUT ICR%(2, K)
            INPUT ICR%(1, K)
            FOR I = 1 TO 2
                INPUT XSD#(I, J, K)
            NEXT I
830     NEXT J
    NEXT K

'-----INPUT DATA INTO REUSABLE FILE-----

840 PRINT #3, NSD%, NRSD%
    PRINT #3, KELM%, IELM%
    FOR K = 1 TO NRSD%
        PRINT #3, NCSD%(K), VCT%(K)
    NEXT K

    FOR K = 1 TO NSD%
        PRINT #3, ICR%(1, K), ICR%(2, K)
        FOR J = 1 TO KELM%
            PRINT #3, XSD#(1, J, K), XSD#(2, J, K)
        NEXT J
```

```
    NEXT K

'-----PRINT DATA TO FILE----

850 PRINT #1,
    PRINT #1, " - - - - - - - Subdomain entry data - - - - - - - "
    FOR K = 1 TO NSD%
        PRINT #1, : PRINT #1,
        PRINT #1, " - - - - - - - - - - - - - - - - - - - - - - - - "
        PRINT #1,
        PRINT #1, SPC(1); "Sub-"; SPC(6); "Elem"; SPC(6); "Elem"; SPC(6);_
        "Rows"; SPC(6); "Cols"
        PRINT #1, SPC(1); "Domain"; SPC(4); "Nodes"; SPC(4); "Int Pt";_
        SPC(5);"Elems"; SPC(5); "Elems"
        PRINT #1, SPC(2); K; SPC(7); KELM%; SPC(7); IELM%; SPC(7);
        ICR%(2, K); SPC(7); ICR%(1, K)
        PRINT #1,
        PRINT #1, SPC(10); " Key nodes of subomain nr "; K
        PRINT #1, ; "Coords";
        FOR I = 1 TO KELM%
            PRINT #1, SPC(8); USING "###"; I;
        NEXT I
        PRINT #1,
        PRINT #1, ; "  X     ";
        FOR I = 1 TO KELM%
            PRINT #1, SPC(5); USING "###.##"; XSD#(1, I, K);
        NEXT I

        PRINT #1,
        PRINT #1, ; "  Y     ";
        FOR I = 1 TO KELM%
            PRINT #1, SPC(5); USING "###.##"; XSD#(2, I, K);
        NEXT I
    NEXT K
    PRINT #1,

    END SUB

'=================SUB EXIST= SUB EXIST ==========

    SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
    SHARED VCT%( )
```

```
    PRINT : PRINT : PRINT : PRINT
    INPUT "Name of existing datafile"; EDFL$

'-----OPEN EXISTING DATA FILE AND READ DATA-----

    OPEN EDFL$ + ".MSH" FOR OUTPUT AS #1
    OPEN EDFL$ + ".TMP" FOR OUTPUT AS #2
    OPEN EDFL$ + ".DAT" FOR INPUT AS #3

    INPUT #3, NSD%, NRSD%
    INPUT #3, KELM%, IELM%

    DIM NCSD%(NRSD%), ICR%(2, NSD%), XSD#(2, KELM%, NSD%)
    DIM VCT%(NRSD%)

    FOR K = 1 TO NRSD%
        INPUT #3, NCSD%(K), VCT%(K)
    NEXT K

    FOR K = 1 TO NSD%
        INPUT #3, ICR%(1, K), ICR%(2, K)
        FOR J = 1 TO KELM%
            INPUT #3, XSD#(1, J, K), XSD#(2, J, K)
        NEXT J
    NEXT K

    PRINT : PRINT : PRINT
    PRINT "Would you like to change the current number "
    PRINT "of intigration points (currently - "; IELM%; ") (Y/N)";
    INPUT ; Y$
    IF Y$ = "N" OR Y$ = "n" THEN 900
    PRINT : PRINT
    INPUT "New number of intigration points"; IELM%

900 END SUB

'==================SUB FLOW1====================

    SUB FLOW1 (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IGLO%, U#( ))

    REDIM XX#(0 TO NROW% + 1, O TO NCOL% + 1, 2)
    REDIM NNGLO%(0 TO NROW% + 1, 0 TO NCOL% + 1)
    REDIM FF#(NROW% + 1, NCOL% + 1)
```

```
      REDIM XX(IGLO%), YY(IGLO%)

      '-----CHANGE ASPECT RATIO  IF NOT A SQUARE THEN SQUARE-----

      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              NNGLO%(I, J) = NGLO%(I, J)
              FOR M = 1 TO 2
                  XX#(I, J, M) = XYGLO#(M, I, J)
              NEXT M
              IF NGLO%(I, J) = 0 THEN 1000
              IJ = IJ + 1
              FF#(I, J) = U#(IJ)
1000      NEXT J
      NEXT I

'-----SET SCREEN-----

1010 SCREEN 0
      CLS
      KEY OFF: DEFINT I-N
      SCREEN 11: F$ = "####.##"
      ASP = .46
      CLS

      XMAX = XX#(1, 1, 1)
      YMAX = XX#(1, 1, 2)
      XMIN = XX#(1, 1, 1)
      YMIN = XX#(1, 1, 2)
      FMAX = FF#(1, 1)
      FMIN = FF#(1, 1)

      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              IF XMAX < XX#(I, J, 1) THEN XMAX = XX#(I, J, 1)
              IF YMAX < XX#(I, J, 2) THEN YMAX = XX#(I, J, 2)
              IF XMIN > XX#(I, J, 1) THEN XMIN = XX#(I, J, 1)
              IF YMIN > XX#(I, J, 2) THEN YMIN = XX#(I, J, 2)
              IF NNGLO%(I, J) = 0 THEN 1020
              IF FMAX < FF#(I, J) THEN FMAX = FF#(I, J)
              IF FMIN > FF#(I, J) THEN FMIN = FF#(I, J)
1020 NEXT J
      NEXT I
```

```
      NN = 1

      LOCATE 17, 25: INPUT "# OF FLOW-LINES"; INTV!
      LOCATE 18, 25: INPUT "CONTONTOUR SKIP INTV"; PP
      LOCATE 19, 25: INPUT "OVERLAP WITH MESH (Y/N)"; msh$

      XL = (XMAX - XMIN)
      YL = (YMAX - YMIN)
      X0 = XMIN - XL / 10
      Y0 = YMIN - YL / 10

      VIEW (51, 1)-(639, 430)

      AA = 538 * ASP / 167

      IF XL / YL > AA THEN YL = XL / AA
      IF XL / YL < AA THEN XL = YL * AA
      XMAX = X0 + 1.2 * XL: YMAX = Y0 + 1.2 * YL
      WINDOW (X0, Y0)-(XMAX, YMAX)
      CLS
      IF msh$ = "Y" OR msh$ = "y" THEN GOTO 1070

'-----FIND BOUNDRY LINES-----

      FOR I = 1 TO (NROW%)
          FOR J = 1 TO (NCOL%)

              IF NNGLO%(I, J) = 0 THEN 1060
              X1 = XX#(I, J, 1)
              Y1 = XX#(I, J, 2)
              IF NNGLO%(I + 1, J + 1) = 0 THEN 1030
              IF NNGLO%(I + 1, J) = 0 OR NNGLO%(I - 1, J) = 0 THEN_
              1030
              IF NNGLO%(I - 1, J + 1) = 0 AND NNGLO%(I - 1, J) < > 0_
              THEN 1030 ELSE 1040

1030          X2 = XX#(I, J + 1, 1)
              Y2 = XX#(I, J + 1, 2)
              IF X2 = 0 THEN 1040
              LINE (X1, Y1)-(X2, Y2)

1040          IF NNGLO%(I + 1, J + 1) = 0 THEN 1050
              IF NNGLO%(I, J + 1) = 0 OR NNGLO%(I, J - 1) = 0_
```

```
                THEN 1050
                IF NNGLO%(I + 1, J - 1) = 0 AND NNGLO%(I, J - 1) <> 0_
                THEN 1050 ELSE 1060

1050        X2 = XX#(I + 1, J, 1)
                Y2 = XX#(I + 1, J, 2)
                IF Y2 = 0 THEN 1060
                LINE (X1, Y1)-(X2, Y2)
1060 NEXT J
     NEXT I

     GOTO 1100

'-----DRAW MESH-----

1070 FOR I = 1 TO (NROW%)
          FOR J = 1 TO (NCOL%)

                IF NNGLO%(I, J) = 0 THEN 1090
                X1 = XX#(I, J, 1)
                Y1 = XX#(I, J, 2)

                IF NNGLO%(I, J + 1) = 0 THEN 1080
                IF J = NCOL% THEN 1080
                X2 = XX#(I, J + 1, 1)
                Y2 = XX#(I, J + 1, 2)

                LINE (X1, Y1)-(X2, Y2)

1080        IF I = NROW% THEN 1090
                IF NNGLO%(I + 1, J) = 0 THEN 1090
                X2 = XX#(I + 1, J, 1)
                Y2 = XX#(I + 1, J, 2)

                LINE (X1, Y1)-(X2, Y2)
1090 NEXT J
     NEXT I


'-----PROGRAM TO CACULATE FLOW LINES-----

1100 OPEN "FLO.TMP" FOR INPUT AS #4
```

```
     INPUT #4, A, B
     INPUT #4, C, D

     INPUT #4, CMIN
     INPUT #4, BX1, BY1
     INPUT #4, BX2, BY2
     INPUT #4, ZERO1, ZERO2

     REDIM X3(100), Y3(100)
     REDIM X4(100), Y4(100)

     REDIM HOLDX!(INTV!)
     REDIM HOLDY!(INTV!)
     REDIM HOLDM!(INTV!)

1110 AA = SQR((BX2 - BX1) ^ 2 + (BY2 - BY1) ^ 2) / (INTV! + 1)
     X1 = BX1: Y1 = BY1
     X2 = BX2: Y2 = BY2
     LINE (X2, Y2)-(X1, Y1)

     L = 0
     PPCNT = 0

1140 PPCNT = PPCNT + 1
     N = 1
     K = 1
     INPUT #4, XINE
     IF XINE = -100000 THEN GOTO 1280

     FOR I = 1 TO 100
          INPUT #4, X3(I), Y3(I)
          IF X3(I) = 0 AND Y3(I) = 0 THEN 1165
          INPUT #4, X4(I), Y4(I)
          IF PPCNT = PP THEN 1150 ELSE 1160
1150 LINE (X3(I), Y3(I))-(X4(I), Y4(I))
1160 N = N + 1
     NEXT I

1165 IF L <> 0 THEN 1170
     IF X1 <> X2 THEN 1166
     FOR I! = (Y1 + AA) TO Y2 STEP AA
          HOLDX!(K) = X1
          HOLDY!(K) = I!
```

```
        K = K + 1
    NEXT I!
    K = 1
    GOTO 1170

1166 IF Y1 < > Y2 THEN 1167
    FOR I! = (X1 + AA) TO X2 STEP AA
        HOLDX!(K) = I!
        HOLDY!(K) = Y1
        K = K + 1
    NEXT I!
    K = 1
    GOTO 1170

1167

1170 IF PPCNT = PP THEN PPCNT = 0
    FOR I! = (SQR(X1 ^ 2 + Y1 ^ 2) + AA) TO (SQR(X2 ^ 2 + Y2 ^ 2))_
    STEP AA

    Z1 = HOLDX!(K)
    Z2 = HOLDY!(K)
    M! = HOLDM!(K)
    IF M! = 0 THEN 1190
    MI! = -1 / M!

1190 FOR BB = 1 TO N - 1

        IF X3(BB) = X4(BB) THEN 1270
1200    M2! = (Y4(BB) - Y3(BB)) / (X4(BB) - X3(BB))
        A# = (X4(BB) - X3(BB)) / 50

        FOR J! = X3(BB) TO X4(BB) STEP A#

            Z3 = J!
            Z4 = M2! * (Z3 - X3(BB)) + Y3(BB)

            IF M2! < > 0 THEN 1202
            MI! = -1 / M!
            X = Z1
            Y = MI! * (X - Z3) + Z4
            GOTO 1207
```

```
1202      IF M! < > 0 THEN 1205
              MI2! = -1 / M2!
              X = Z1
              Y = MI2! * (X - Z1) + Z2
              GOTO 1207

1205      IF M! = M2! THEN 1260
              MI! = -1 / M!
              MI2! = -1 / M2!
              X = (Z4 - Z2 + MI! * Z1 - MI2! * Z3) / (MI! - MI2!)
              Y = MI! * (X - Z1) + Z2

1207      LEGTH1! = SQR((Z1 - X) ^ 2 + (Z2 - Y) ^ 2)
              LEGTH2! = SQR((Z3 - X) ^ 2 + (Z4 - Y) ^ 2)

              IF J! = X3(BB) THEN 1220

              IF (LEGTH1! + LEGTH2!) > (LEG1! + LEG2!) THEN 1220

              XSS = X
              YSS = Y
              MS! = M2!

              ZS3 = Z3
              ZS4 = Z4

1220      LEG1! = LEGTH1!
              LEG2! = LEGTH2!

          NEXT J!
      NEXT BB

      IF Z1 < XSS AND XSS < ZS3 THEN 1230
      IF Z1 > XSS AND XSS > ZS3 THEN 1230 ELSE 1240

1230 LINE (Z1, Z2)-(XSS, YSS)
      LINE (ZS3, ZS4)-(XSS, YSS)
      GOTO 1250

1240 LINE (ZS3, ZS4)-(Z1, Z2)

1250 HOLDX!(K) = ZS3
      HOLDY!(K) = ZS4
```

```
    HOLDM!(K) = MS!

1260 K = K + 1

1270 NEXT I!
    L = L + 1
    GOTO 1140

1280 INPUT #4, CTR
    INPUT #4, X3, Y3
    INPUT #4, X4, Y4
    LINE (X3, Y3)-(X4, Y4)

    K = 1
    FOR I! = (Y1 + AA) TO Y2 STEP AA
        Z1 = HOLDX!(K)
        Z2 = HOLDY!(K)
        M! = HOLDM!(K)
        MI! = -1 / M!

        IF X4 = X3 THEN 1285 ELSE 1290
1285 X = X3
        Y = MI! * (X - Z1) + Z2
        GOTO 1295

1290 M2! = (Y4 - Y3) / (X4 - X3)
        X = (Z4 - Z2 + MI! * Z1 - M2! * Z3) / (M! - M2!)
        Y = MI! * (X - Z1) + Z2
1295 LINE (Z1, Z2)-(X, Y)
        K = 1 + K
    NEXT I!

    CLOSE #4
    LOCATE 29, 5: INPUT "WANT TO REDRAW WITH NEW FLOWLINE_
    INTERVAL (Y/N)"; RED$
    IF RED$ = "Y" OR RED$ = "y" THEN 1010

    END SUB

    DEFSNG I-N
'================SUB  MESH================

    SUB MESH (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IGLO%)
```

```
      SHARED PSI#( ), DPSI#( )
      SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )
      SHARED XXI#, EETA#
      SHARED VCT%( )

      DIM PSI#(KELM%), DPSI#(2, KELM%)

      L% = 0

'-----ASSUME LINEAR SUBDOMAINS AND ELEMENTS;------
'     ADJUST IF QUADRATIC

      NLQ% = 1
      IF KELM% <> 4 THEN NLQ% = 2

'-----ZERO OUT THE GLOBAL X-Y COORD AND NODE NR ARRAYS-----

      FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
              NGLO% = 0
              FOR M = 1 TO 2
                  XYGLO#(M, I, J) = 0#
              NEXT M
          NEXT J
      NEXT I

'-----LOOP ON SUBDOMAINS TO CALCULATE GLOBAL X-Y COORDS-----

      K% = 0
      IO% = 0
      JO% = 0
      ISE% = 1
      K1% = 1

'-----LOOP ON ROWS OF SUBDOMAINS; SET LOOP PARAMETERS;------
'     CALC ETA INCREMENT

      FOR LL% = 1 TO NRSD%
          IX% = ISE%
          K1% = K% + 1
          NRSE% = NLQ% * ICR%(2, K1%)
          ISE% = ISE% + NRSE%
```

```
          DNR# = NRSE%
          DETA# = 2# / DNR#
          JSE% = 1

'-----LOOP ON SUBDOMAINS IN ROW; SET LOOP PARAMETERS; SET XI-----
'     AND ETA TO INITIAL VALUES; CALCULATE XI INCREMENT

          FOR KK% = 1 TO NCSD%(LL%)
               K% = K% + 1
               XIO# = -1#
               ETAO# = -1#
               IF LL% = 1 THEN GOTO 1300
               IF KK% = 1 THEN JSE% = VCT%(LL%)
1300       JY% = JSE%
               NCSE% = NLQ% * ICR%(1, K%)
               JSE% = JSE% + NCSE%
               DNC# = NCSE%
               DXI# = 2# / DNC#

'-----RE-ZERO OUT GLOBAL X-Y COORD ARRAY BY SUBDOMAIN-----

               FOR I = IX% TO ISE%
                    FOR J = JY% TO JSE%
                         FOR M = 1 TO 2
                              XYGLO#(M, I, J) = 0#
                         NEXT M
                    NEXT J
               NEXT I

'-----CALCULATE XI-ETA COORDS BY NODE ROW WITHIN THE-----
'     MASTER SUBDOMAIN

               XXI# = XIO# - DXI#
               FOR I = IX% TO ISE%
                    FOR J = JY% TO JSE%
                         XXI# = XXI# + DXI#
                         EETA# = ETAO#

'-----MAP XI-ETA COORDS INTO X-Y COORDS ON THE PHYSICAL-----
'     SUBDOMAIN AND IDENTIFY ANY NODE AT THE X-Y ORIGIN
'     ON THE PHYSICAL DOMAIN

                              CALL SHAPE(L%)
```

```
                        FOR M = 1 TO 2
                            FOR LA% = 1 TO KELM%
                                XYGLO#(M, I, J) = XYGLO#(M, I, J) +_
                                XSD#(M, LA%, K%) * PSI#(LA%)
                            NEXT LA%
                        NEXT M
                        XPY# = ABS(XYGLO#(1, I, J)) +_
                        ABS(XYGLO#(2, I, J))
                        IF XPY# > .001 THEN GOTO 1310
                        IO% = I
                        JO% = J
1310            NEXT J

'-----ADJUST STARTING XI-ETA# COORDS FOR NEXT ROW OF NODES-----
'    IN THE SUBDOMAIN

                    XXI# = XIO# - DXI#
                    ETAO# = ETAO# + DETA#
                NEXT I
            NEXT KK%
        NEXT LL%

'-----ZERO OUT CENTER X-Y COORDS FOR THE 8 NODE-----
'    QUADRATIC ELEMENTS

    IF KELM% <> 8 THEN GOTO 1320
    FOR I = 1 TO (NROW% / 2)
        FOR J = 1 TO (NCOL% / 2)
            FOR K% = 1 TO 2
                XYGLO#(K%, 2 * I, 2 * J) = 0#
            NEXT K%
        NEXT J
    NEXT I


'-----ASSIGN GLOBAL NODE NRS TO NON-ZERO ENTRIES IN GLOBAL-----
'    X-Y COORD ARRAY

1320 IGLO% = 0
    FOR I = 1 TO NROW%
        FOR J = 1 TO NCOL%
            IF I = IO% AND J = JO% THEN GOTO 1330
```

```
            XPY# = ABS(XYGLO#(1, I, J)) + ABS(XYGLO#(2, I, J))
            IF XPY# < .001 GOTO 1340
1330    IGLO% = IGLO% + 1
            NGLO%(I, J) = IGLO%
1340 NEXT J
    NEXT I

    END SUB

'================SUB PLOT2D====================

    SUB PLOT2D (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IGLO%,_
    PAUSE$)

    SHARED NRE%
    DIM XX#(NROW% + 1, NCOL% + 1, 2)
    DIM NNGLO%(NROW% + 1, NCOL% + 1)

'-----SET SCREEN-----

    KEY OFF: DEFINT I-N
    SCREEN 11: F$ = "####.##"
    ASP = .46

'-----CHANGE ASPECT RATIO IF NOT SQUARE THEN SQUARE------

    LOCATE 16, 25
    FOR I = 1 TO NROW%
        FOR J = 1 TO NCOL%
            NNGLO%(I, J) = NGLO%(I, J)
            FOR M = 1 TO 2
                XX#(I, J, M) = XYGLO#(M, I, J)
            NEXT M
        NEXT J
    NEXT I

    XMAX = XX#(1, 1, 1)
    YMAX = XX#(1, 1, 2)
    XMIN = XX#(1, 1, 1)
    YMIN = XX#(1, 1, 2)

    FOR I = 2 TO NROW%
        FOR J = 2 TO NCOL%
```

```
            IF XMAX < XX#(I, J, 1) THEN XMAX = XX#(I, J, 1)
            IF YMAX < XX#(I, J, 2) THEN YMAX = XX#(I, J, 2)
            IF XMIN > XX#(I, J, 1) THEN XMIN = XX#(I, J, 1)
            IF YMIN > XX#(I, J, 2) THEN YMIN = XX#(I, J, 2)
        NEXT J
    NEXT I

    CLS
    XL = (XMAX - XMIN)
    YL = (YMAX - YMIN)
    X0 = XMIN - XL / 10
    Y0 = YMIN - YL / 10

    VIEW (51, 1)-(639, 350)

    AA = 538 * ASP / 167
    IF XL / YL > AA THEN YL = XL / AA
    IF XL / YL < AA THEN XL = YL * AA

    XMAX = X0 + 1.2 * XL
    YMAX = Y0 + 1.2 * YL
    WINDOW (X0, Y0)-(XMAX, YMAX)

'-----DRAW ELEMENTS-----

    CLS
    FOR I = 1 TO (NROW%)
        FOR J = 1 TO (NCOL%)

            X1 = XX#(I, J, 1)
            Y1 = XX#(I, J, 2)
            IF NNGLO%(I, J) = 0 THEN 1410
            IF NNGLO%(I, J + 1) = 0 THEN 1400

            IF J = NCOL% THEN 1400
            X2 = XX#(I, J + 1, 1)
            Y2 = XX#(I, J + 1, 2)

            LINE (X1, Y1)-(X2, Y2)

1400        IF I = NROW% THEN 1410
            IF NNGLO%(I + 1, J) = 0 THEN 1410
            X2 = XX#(I + 1, J, 1)
```

```
            Y2 = XX#(I + 1, J, 2)

            LINE (X1, Y1)-(X2, Y2)

1410 NEXT J
     NEXT I
     NX% = NROW% * NCOL%
     LOCATE 27, 14: PRINT "Nr of nodes is"; IGLO%; ",  Nr of elems is";_
     NRE%
     LOCATE 28, 14: PRINT "Nr of (real + blank) nodes is"; NX%
     LOCATE 29, 14: INPUT "Is the above mesh correct (Y/N)"; PAUSE$
     SCREEN 0
     CLS

     END SUB

     DEFSNG I-N
'==========SUB PROUT==========

     SUB PROUT (NROW%, NCOL%, NGLO%( ), XYGLO#( ), IGLO%, U#( ))

     SHARED NCOLE%(), NRE%

'-----PRINT DATA TO A FOR,ATED DATA FILE-----

     PRINT #1,
     PRINT #1, : PRINT #1, SPC(12); " * * * * RESULTS * * * *"
     PRINT #1, : PRINT #1, SPC(5); " Nr of nodes is"; IGLO%; "   Nr of elems_
     is"; NRE%
     PRINT #1, : PRINT #1,
     PRINT #1, "node nr"; SPC(6); "x-coord"; SPC(8); "y-coord"; SPC(11); "U":
     PRINT
     PRINT #1,

     IG% = 0
     FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
               IF NGLO%(I, J) = 0 GOTO 1500
               IG% = IG% + 1
               PRINT #1, USING "#####"; NGLO%(I, J);
               FOR K = 1 TO 2
                    PRINT #1, SPC(7); USING "###.####"; XYGLO#(K, I, J);
               NEXT K
```

```
                PRINT #1, SPC(6); USING "###.####"; U#(IG%)
1500 NEXT J
     NEXT I

'-----PRINT DATA TO UNFORMATED DATA FILE-----

     PRINT #2, NROW%, NCOL%, IGLO%

     IG% = 0
     FOR I = 1 TO NROW%
          FOR J = 1 TO NCOL%
               IF NGLO%(I, J) = 0 GOTO 1510
               IG% = IG% + 1
               PRINT #2, NGLO%(I, J)
               FOR K = 1 TO 2
                    PRINT #2, XYGLO#(K, I, J)
               NEXT K
               PRINT #2, U#(IG%)
1510 NEXT J
     NEXT I

     END SUB

'==================SUB RBC====================

     SUB RBC

     SHARED NRB%, KBC%( ), VBC#( ), ISIDE%( ), NRBE%( )

     DIM KBC%(25), VBC#(2, 25), ISIDE%(25), NRBE%(3, 25)

'-----ENTER BOUNDARY CONDITION DATA BY BOUNDARY STRIPS-----

1600 PRINT
     INPUT " Enter the nr of boundary strips, no more than 25"; NRB%
     IF NRB% <= 25 GOTO 1610
     PRINT : PRINT "Too many boundary strips": PRINT
     GOTO 1600

1610 CLS : PRINT
     PRINT " Start with the 1st boundary strip and enter in succession for each"
     PRINT SPC(6); "- kind of BC (1 = essential; 2 = natural; 3 = mixed)"
     PRINT SPC(6); "- value of BC (U:type 1; K*DU/DN:type 2 positive outward;"
```

```
      PRINT SPC(6); "  H and UE:type 3 where K*DU/DN = H*(U-UE); H is_
      positive"
      PRINT SPC(6); "- elem side nr wherE BC is applied (1 for elem nodes 1-2,"
      PRINT SPC(6); "  2 for 2-3, 3 for 3-4, or 4 for 4-1)"
      PRINT SPC(6); "- lowest elem nr"
      PRINT SPC(6); "- highest elem nr"
      PRINT SPC(6); "- increment in elem nrs as they vary lowest to highest":
      PRINT
      NB% = 0

      CALL BCDAT(NB%)

'-----CORRECT THE BC ENTRIES BY STRIP, AS NEEDED-----

1620 PRINT : INPUT " Are all BC data correct? y/n"; IY$
      IF IY$ = "Y" OR IY$ = "y" GOTO 1640
1630 PRINT : INPUT " Enter nr of the boundary strip to be corrected"; NB%
      PRINT : PRINT " Re-enter the 6 or 7 data entries for this strip in the same_
      sequence as before": PRINT

      CALL BCDAT(NB%)

      GOTO 1620

'-----DISPLAY THE BC DATA-----

1640 PRINT : INPUT " Want the BC data displayed? y/n"; IY$
      IF IY$ = "N" OR IY$ = "n" GOTO 1650

      PRINT : PRINT " Boundary Condition Data - - - - - - - - - - - -"
      PRINT : PRINT " Bound"; SPC(2); "Type"; SPC(5); "Value of Cond";_
      SPC(5);"Side"; SPC(7); "Element nrs"
      PRINT " Strip"; SPC(2); "Cond"; SPC(3); "1st Val"; SPC(3); "2nd Val";_
      SPC(4); "Nr"; SPC(5); "Low"; SPC(3); "High"; SPC(3); "Incr"

      FOR I = 1 TO NRB%
            PRINT USING "###"; I; SPC(4);
            PRINT USING "####"; KBC%(I);
            FOR J = 1 TO 2
                  PRINT ; SPC(4); USING "###.##"; VBC#(J, I);
            NEXT J
            PRINT SPC(4); USING "###"; ISIDE%(I);
            FOR J = 1 TO 3
```

```
            PRINT SPC(4); USING "###"; NRBE%(J, I);
        NEXT J
        PRINT
    NEXT I

    PRINT : INPUT " Are these correct? y/n"; IY$
    IF IY$ = "N" OR IY$ = "n" GOTO 1630

'-----PUT THE BC DATA IN THE OUTPUT-----

1650 PRINT #1, : PRINT #1, " Boundary Condition Data - - - - - - - - - - - - -"
    PRINT #1, : PRINT #1, " Bound"; SPC(2); "Type"; SPC(5); "Value of_
    Cond"; SPC(5); "Side"; SPC(7); "Element nrs"
    PRINT #1, " Strip"; SPC(2); "Cond"; SPC(3); "1st Val"; SPC(3); "2nd Val";
    SPC(4); "Nr"; SPC(5); "Low"; SPC(3); "High"; SPC(3); "Incr"

    FOR I = 1 TO NRB%
        PRINT #1, USING "###"; I; SPC(4);
        PRINT #1, USING "####"; KBC%(I);
        FOR J = 1 TO 2
            PRINT #1, ; SPC(4); USING "###.##"; VBC#(J, I);
        NEXT J
        PRINT #1, SPC(4); USING "###"; ISIDE%(I);
        FOR J = 1 TO 3
            PRINT #1, SPC(4); USING "###"; NRBE%(J, I);
        NEXT J
        PRINT #1,
    NEXT I

    END SUB

'==================SUB SETINT==================

    SUB SETINT

    SHARED XIETA#( ), WT#( )
    SHARED NSD%, NRSD%, NCSD%( ), KELM%, IELM%, ICR%( ), XSD#( )

    DIM XIETA#(2, 9), WT#(9)

    IF IELM% = 4 GOTO 1700
    IF IELM% = 9 GOTO 1710
```

```
'-----THE FOLLOWING DEFINES INTEGRATION POINTS AND-----
'     WEIGHTS FOR QUADRILATERAL ELEMENTS

'-----FOUR POINT QUADRATURE-----

1700 A4# = 1# / SQR(3#)
     XIETA#(1, 1) = -A4#
     XIETA#(2, 1) = -A4#
     XIETA#(1, 2) = A4#
     XIETA#(2, 2) = -A4#
     XIETA#(1, 3) = -A4#
     XIETA#(2, 3) = A4#
     XIETA#(1, 4) = A4#
     XIETA#(2, 4) = A4#
     FOR I = 1 TO 4
          WT#(I) = 1#
     NEXT I

     EXIT SUB

'-----NINE POINT QUADRATURE-----

1710 A9# = SQR(.6#)
     XIETA#(1, 1) = -A9#
     XIETA#(2, 1) = -A9#
     XIETA#(1, 2) = 0#
     XIETA#(2, 2) = -A9#
     XIETA#(1, 3) = A9#
     XIETA#(2, 3) = -A9#
     XIETA#(1, 4) = -A9#
     XIETA#(2, 4) = 0#
     XIETA#(1, 5) = 0#
     XIETA#(2, 5) = 0#
     XIETA#(1, 6) = A9#
     XIETA#(2, 6) = 0#
     XIETA#(1, 7) = -A9#
     XIETA#(2, 7) = A9#
     XIETA#(1, 8) = 0#
     XIETA#(2, 8) = A9#
     XIETA#(1, 9) = A9#
     XIETA#(2, 9) = A9#
     WT#(1) = 25# / 81#
```

```
      WT#(2) = 40# / 81#
      WT#(3) = WT#(1)
      WT#(4) = WT#(2)
      WT#(5) = 64# / 81#
      WT#(6) = WT#(2)
      WT#(7) = WT#(1)
      WT#(8) = WT#(2)
      WT#(9) = WT#(1)

      END SUB

'====================SUB SHAPE================

      SUB SHAPE (L%)

      SHARED PSI#(), DPSI#()
      SHARED NSD%, NRSD%, NCSD%(), KELM%, IELM%, ICR%(), XSD#()
      SHARED XXI#, EETA#
      SHARED XIETA#(), WT#()

'-----CALCULATE THE VALUES OF THE SHAPE FUNCTIONS AND THEIR-----
'     DERIVATIVES AT MASTER SUBDOMAIN NODES FOR MESH
'     GENERATION AND MAPPING (L=0) AND AT APPROPRIATE
'     INTEGRATION POINTS WITHIN ELEMENTS (KO=1).

      LO% = L%
      XI# = XXI#
      ETA# = EETA#
      IF LO% = 0 GOTO 1800
      XI# = XIETA#(1, LO%)
      ETA# = XIETA#(2, LO%)
1800 IF KELM% = 4 GOTO 1810
      IF KELM% = 8 GOTO 1820
      IF KELM% = 9 GOTO 1830

'-----LINEAR SHAPE FUNCTIONS AND DERIVATIVES -----
'     4 NODE QUAD ELEMENT

1810 PSI#(1) = .25# * (1# - XI#) * (1# - ETA#)
      PSI#(2) = .25# * (1# + XI#) * (1# - ETA#)
      PSI#(3) = .25# * (1# + XI#) * (1# + ETA#)
      PSI#(4) = .25# * (1# - XI#) * (1# + ETA#)
```

```
      IF LO% = 0 THEN EXIT SUB

      DPSI#(1, 1) = -.25# * (1# - ETA#)
      DPSI#(2, 1) = -.25# * (1# - XI#)
      DPSI#(1, 2) = .25# * (1# - ETA#)
      DPSI#(2, 2) = -.25# * (1# + XI#)
      DPSI#(1, 3) = .25# * (1# + ETA#)
      DPSI#(2, 3) = .25# * (1# + XI#)
      DPSI#(1, 4) = -.25# * (1# + ETA#)
      DPSI#(2, 4) = .25# * (1# - XI#)

      EXIT SUB

'-----QUADRATIC SHAPE FUNCTIONS AND DERIVATIVES -----
'     8 NODE QUAD ELEMENT

1820 PSI#(1) = .25# * (1# - XI#) * (1# - ETA#) * (-1# - XI# - ETA#)
      PSI#(2) = .25# * (1# + XI#) * (1# - ETA#) * (-1# + XI# - ETA#)
      PSI#(3) = .25# * (1# + XI#) * (1# + ETA#) * (-1# + XI# + ETA#)
      PSI#(4) = .25# * (1# - XI#) * (1# + ETA#) * (-1# - XI# + ETA#)
      PSI#(5) = .5# * (1# - XI# ^ 2#) * (1# - ETA#)
      PSI#(6) = .5# * (1# + XI#) * (1# - ETA# ^ 2#)
      PSI#(7) = .5# * (1# - XI# ^ 2#) * (1# + ETA#)
      PSI#(8) = .5# * (1# - XI#) * (1# - ETA# ^ 2#)

      IF LO% = 0 THEN EXIT SUB

      DPSI#(1, 1) = .25# * (1# - ETA#) * (2# * XI# + ETA#)
      DPSI#(2, 1) = .25# * (1# - XI#) * (XI# + 2# * ETA#)
      DPSI#(1, 2) = .25# * (1# - ETA#) * (2# * XI# - ETA#)
      DPSI#(2, 2) = .25# * (1# + XI#) * (-XI# + 2# * ETA#)
      DPSI#(1, 3) = .25# * (1# + ETA#) * (2# * XI# + ETA#)
      DPSI#(2, 3) = .25# * (1# + XI#) * (XI# + 2# * ETA#)
      DPSI#(1, 4) = .25# * (1# + ETA#) * (2# * XI# - ETA#)
      DPSI#(2, 4) = .25# * (1# - XI#) * (-XI# + 2# * ETA#)
      DPSI#(1, 5) = -XI# * (1# - ETA#)
      DPSI#(2, 5) = -.5# * (1# - XI# ^ 2#)
      DPSI#(1, 6) = .5# * (1# - ETA# ^ 2#)
      DPSI#(2, 6) = -ETA# * (1# + XI#)
      DPSI#(1, 7) = -XI# * (1# + ETA#)
      DPSI#(2, 7) = .5# * (1# - XI# ^ 2#)
      DPSI#(1, 8) = -.5# * (1# - ETA# ^ 2#)
      DPSI#(2, 8) = -ETA# * (1# - XI#)
```

```
EXIT SUB

'-----BIQUADRATIC SHAPE FUNCTIONS AND DERIVATIVES ------
'     9 NODE BIQUAD ELEMENT

1830 PSI#(1) = .25# * XI# * ETA# * (XI# - 1#) * (ETA# - 1#)
     PSI#(2) = .25# * XI# * ETA# * (XI# + 1#) * (ETA# - 1#)
     PSI#(3) = .25# * XI# * ETA# * (XI# + 1#) * (ETA# + 1#)
     PSI#(4) = .25# * XI# * ETA# * (XI# - 1#) * (ETA# + 1#)
     PSI#(5) = .5# * (1# - XI# ^ 2#) * (ETA# ^ 2# - ETA#)
     PSI#(6) = .5# * (XI# ^ 2# + XI#) * (1# - ETA# ^ 2#)
     PSI#(7) = .5# * (1# - XI# ^ 2#) * (ETA# ^ 2# + ETA#)
     PSI#(8) = .5# * (XI# ^ 2# - XI#) * (1# - ETA# ^ 2#)
     PSI#(9) = (1# - XI# ^ 2#) * (1# - ETA# ^ 2#)

     IF LO% = 0 THEN EXIT SUB

     DPSI#(1, 1) = .25# * (2# * XI# - 1#) * (ETA# - 1#) * ETA#
     DPSI#(2, 1) = .25# * XI# * (XI# - 1#) * (2# * ETA# - 1#)
     DPSI#(1, 2) = .25# * (2# * XI# + 1#) * ETA# * (ETA# - 1#)
     DPSI#(2, 2) = .25# * XI# * (XI# + 1#) * (2# * ETA# - 1#)
     DPSI#(1, 3) = .25# * (2# * XI# + 1#) * ETA# * (ETA# + 1#)
     DPSI#(2, 3) = .25# * XI# * (XI# + 1#) * (2# * ETA# + 1#)
     DPSI#(1, 4) = .25# * (2# * XI# - 1#) * ETA# * (ETA# + 1#)
     DPSI#(2, 4) = .25# * XI# * (XI# - 1#) * (2# * ETA# + 1#)
     DPSI#(1, 5) = -XI# * ETA# * (ETA# - 1#)
     DPSI#(2, 5) = .5# * (1# - XI# ^ 2#) * (2# * ETA# - 1#)
     DPSI#(1, 6) = .5# * (2# * XI# + 1#) * (1# - ETA# ^ 2#)
     DPSI#(2, 6) = -ETA# * XI# * (XI# + 1#)
     DPSI#(1, 7) = -XI# * ETA# * (ETA# + 1#)
     DPSI#(2, 7) = .5# * (1# - XI# ^ 2#) * (2# * ETA# + 1#)
     DPSI#(1, 8) = .5# * (2# * XI# - 1#) * (1# - ETA# ^ 2#)
     DPSI#(2, 8) = -ETA# * XI# * (XI# - 1#)
     DPSI#(1, 9) = -2# * XI# * (1# - ETA# ^ 2#)
     DPSI#(2, 9) = -2# * ETA# * (1# - XI# ^ 2#)

     END SUB

'==================SUB SIZE==================

     SUB SIZE (NROW%, NCOL%)
```

```
      SHARED NSD%, NRSD%, NCSD%(), KELM%, IELM%, ICR%(), XSD#()
      SHARED NCOLE%(), NRE%
      SHARED VCT%()

'-----ASSUME LINEAR SUBDOMAINS AND ELEMENTS;-----
'     ADJUST IF QUADRATIC

      NLQ% = 1
      IF KELM% <> 4 THEN NLQ% = 2
      DIM NCOLE%(NRSD%)

'-----CALCULATE NR OF ELEMS AND MAX NR OF NODES-----
'     PER ROW/COL IN THE DOMAIN

      FOR I = 1 TO NRSD%
          NCOLE%(I) = 0
      NEXT I

      K% = 0
      NROW% = 0
      NRE% = 0
      FOR I = 1 TO NRSD%
          NCOLE%(I) = VCT%(I) - 1
          FOR J = 1 TO NCSD%(I)
              K% = K% + 1
              NRE% = NRE% + ICR%(1, K%) * ICR%(2, K%)
              IF J <> 1 THEN GOTO 1900
              NROW% = NROW% + NLQ% * ICR%(2, K%)
1900      NCOLE%(I) = NCOLE%(I) + NLQ% * ICR%(1, K%)
          NEXT J
      NEXT I
      NCOL% = 1
      FOR I = 1 TO NRSD%
          IF NCOL% < NCOLE%(I) THEN NCOL% = NCOLE%(I)
      NEXT I
      NROW% = NROW% + 1
      NCOL% = NCOL% + 1

      END SUB

'==================SUB SOLVE==================

      SUB SOLVE (IGLO%, IBW%, GF#( ), GK#( ), U#( ))
```

```
      NN2% = IGLO%
      IBB% = IBW%
      IBH% = (IBB% / 2 - .5)
      IBH1% = IBH% + 1
      IBH2% = IBH% + 2

'-----DIVIDE PIVOT ROW ENTRIES BY PIVOT-----

      FOR ICOL% = 1 TO NN2%
          PIVOT# = GK#(ICOL%, IBH1%)

'-----STOP PROGRAM IF MATRIX IS SINGULAR-----

          IF ABS(PIVOT#) < .00001 THEN GOTO 2000 ELSE 2010
2000 PRINT : PRINT " The stiffness matrix is singular": PRINT
          STOP

2010 FOR J = 1 TO IBB%
               GK#(ICOL%, J) = (GK#(ICOL%, J) / PIVOT#)
          NEXT J

          GF#(ICOL%) = (GF#(ICOL%) / PIVOT#)

'-----SUBTRACT FROM EACH ROW THE PIVOT ROW MULTIPLIED BY-----
'     THE ROW'S PIVOT COLUMN ENTRY

          FOR I = 1 TO NN2%
               IW% = I - ICOL%
               IF IW% < 1 OR IW% > IBH% GOTO 2030
               JCOL% = IBH1% - IW%
               TEMP# = GK#(I, JCOL%)
               FOR J = 1 TO IBB%
                    JP% = J + IW%
                    IF JP% > IBB% GOTO 2020
                    GK#(I, J) = GK#(I, J) - GK#(ICOL%, JP%) * TEMP#
2020           NEXT J
               GF#(I) = GF#(I) - GF#(ICOL%) * TEMP#

               LOCATE 10, 26: PRINT "*****************************"
               LOCATE 11, 26: PRINT "*    SOLVING MATRIX       *"
               LOCATE 12, 26: PRINT "*         "; TIME$; "       *"
               LOCATE 13, 26: PRINT "*****************************"
```

```
2030 NEXT I
     NEXT ICOL%

'-----BACK SUBSTITUTION-----

     FOR I = 1 TO NN2%
          IB% = NN2% - I + 1
          FOR J = IBH2% TO IBB%
               IJ% = IB% + J - IBH1%
               IF IJ% > NN2% GOTO 2040
               GF#(IB%) = GF#(IB%) - GK#(IB%, J) * GF#(IJ%)
2040 NEXT J
     NEXT I

     FOR I = 1 TO NN2%
          U#(I) = GF#(I)
     NEXT I

END SUB

'==================SUB ZSORT================

     SUB ZSORT

'-----THIS PROGRAM SORTS CONTOUR DATA FILE-----

     OPEN "CONTR.TMP" FOR INPUT AS #4
     OPEN "FLO.TMP" FOR OUTPUT AS #5
     DIM A!(100), B!(100), C!(100), D!(100), K!(200), L!(200)

     INPUT #4, A!, B!
     INPUT #4, C!, D!
     PRINT #5, A!, B!
     PRINT #5, C!, D!
     INPUT #4, E!
     PRINT #5, E!

     FOR I = 1 TO 3
          INPUT #4, A!, B!
          PRINT #5, A!, B!
     NEXT I
```

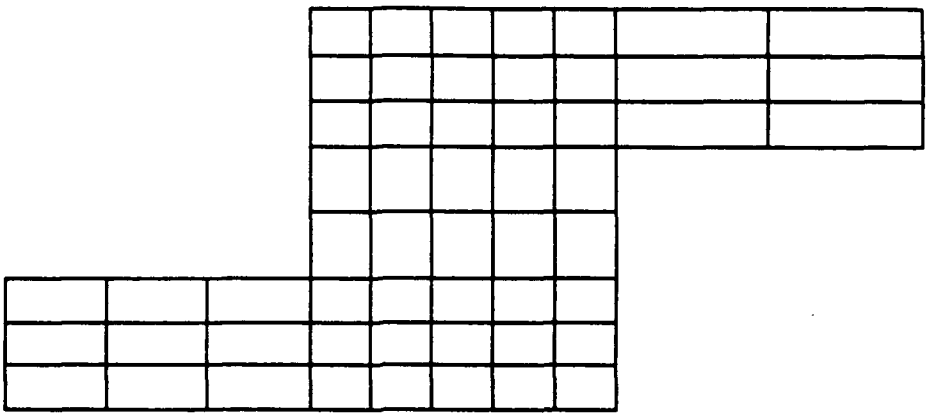
```
     CLS
2100 N = 1
     INPUT #4, E!
     PRINT #5, E!

     IF E = -100000 THEN 2190

     FOR I = 1 TO 100
          INPUT #4, A!(I), B!(I)
          IF A!(I) = 0 AND B!(I) = 0 THEN 2110
          INPUT #4, C!(I), D!(I)
          N = N + 1
     NEXT I

2110 FOR I = 1 TO N - 1
          LET L!(I) = A!(I)
          LET L!(I + (N - 1)) = C!(I)
          LET K!(I) = B!(I)
          LET K!(I + (N - 1)) = D!(I)
     NEXT I

     FOR I = 1 TO (2 * N - 2)
          FOR J = 1 TO (I - 1)
               IF L!(I) = L!(J) AND K!(I) = K!(J) THEN 2120
          NEXT J
          FOR J = I + 1 TO (2 * N - 2)
               IF L!(I) = L!(J) AND K!(I) = K!(J) THEN 2120
          NEXT J
          AA! = L!(I)
          BB! = K!(I)
          GOTO 2130
2120 NEXT I

2130 FOR I = 1 TO 40
          FOR J = 1 TO N - 1
               IF A!(J) = AA! AND B!(J) = BB! THEN 2140 ELSE 2150
2140       AA! = C!(J)
               BB! = D!(J)
               PRINT #5, A!(J), B!(J)
               PRINT #5, C!(J), D!(J)
               D(J) = RND
               GOTO 2180
2150       IF C!(J) = AA! AND D!(J) = BB! THEN 2160 ELSE 2170
```

```
2160        AA! = A!(J)
                  BB! = B!(J)
                  PRINT #5, C!(J), D!(J)
                  PRINT #5, A!(J), B!(J)
                  B!(J) = RND
                  GOTO 2180
2170 NEXT J
2180 NEXT I

     LOCATE 10, 26: PRINT "******************************"
     LOCATE 11, 26: PRINT "*   SORTING CONTOUR DATA   *"
     LOCATE 12, 26: PRINT "*          "; TIME$; "        *"
     LOCATE 13, 26: PRINT "******************************"

     PRINT #5, 0, 0
     GOTO 2100

2190 INPUT #4, CTR
     PRINT #5, CTR

     INPUT #4, A!, B!
     INPUT #4, C!, D!
     PRINT #5, A!, B!
     PRINT #5, C!, D!

     CLOSE #4, #5

     END SUB
```

APPENDIX B

EXAMPLE PROBLEM - RESULTS

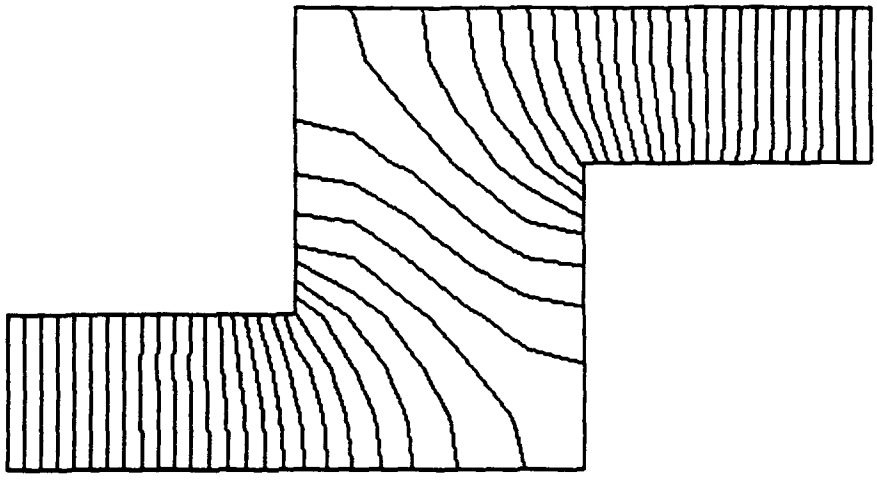
Finite Element Mesh



```
Nr of nodes is 74 ,  Nr of elems is 55
Nr of (real + blank) nodes is 99
Is the above mesh correct (Y/N)? 
```

Figure B.1
Finite Element Mesh

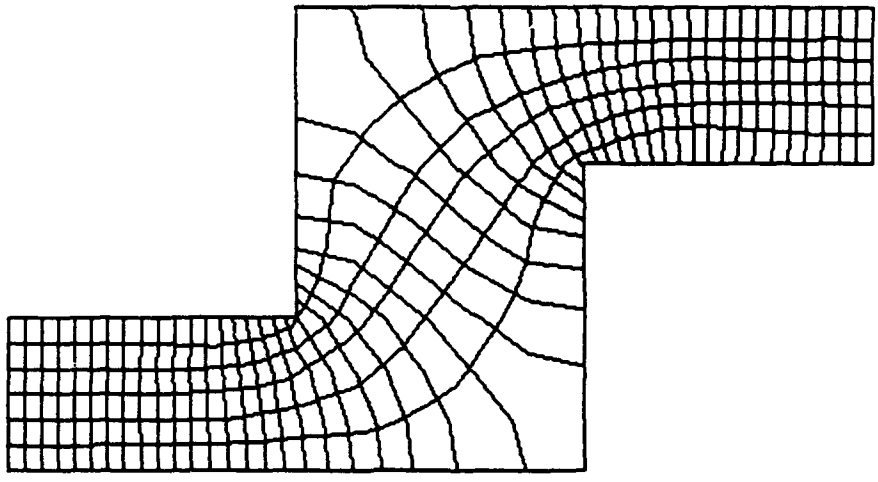
Contour Graph



```
BOUNDARY CONDITIONS, MIN =  15 , MAX =  30
CONTOUR INTERVAL =   0.3000
WANT TO REDRAW WITH NEW CONTOUR INTERVAL (Y/N)?
```

Figure B.2
Contour Graph

Flow Line Graph



WANT TO REDRAW WITH NEW FLOWLINE INTERVAL (Y/N)? ▌

Figure B.3
Flow Line Graph

Data File "Example.dat"

Data File "Example Dat"

```
Preliminary Data
 5                3
 4                4
 2                1
 1                4
 2                4
Subdomain 1
 3                3
 1                1
 3                1
 3                2
 1                2
Subdomain 2
 5                3
 3                1
 5                1
 5                2
 3                2
Subdomain 3
 5                2
 3                2
 5                2
 5                3
 3                3
Subdomain 4
 5                3
 3                3
 5                3
 5                4
 3                4
Sumdomain 5
 2                3
 5                3
 7                3
 7                4
 5                4
```

Data File "Example.msh"

Data File "Example.msh"

```
- - - - - - - - Subdomain entry data  - - - - - - - -

 - - - - - - - - For file example - - - - - - - -
 Sub-       Elem       Elem       Rows       Cols
 Domain     Nodes     Int Pt      Elems      Elems
   1          4          4          3          3

           Key nodes of subdomain nr  1
Coords           1          2          3          4
  X           1.00       3.00       3.00       1.00
  Y           1.00       1.00       2.00       2.00
 - - - - - - - - For file example - - - - - - - -
 Sub-       Elem       Elem       Rows       Cols
 Domain     Nodes     Int Pt      Elems      Elems
   2          4          4          3          5

           Key nodes of subdomain nr  2
Coords           1          2          3          4
  X           3.00       5.00       5.00       3.00
  Y           1.00       1.00       2.00       2.00
 - - - - - - - - For file example - - - - - - - -
 Sub-       Elem       Elem       Rows       Cols
 Domain     Nodes     Int Pt      Elems      Elems
   3          4          4          2          5

           Key nodes of subdomain nr  3
Coords           1          2          3          4
  X           3.00       5.00       5.00       3.00
  Y           2.00       2.00       3.00       3.00
 - - - - - - - - For file example - - - - - - - -
 Sub-       Elem       Elem       Rows       Cols
 Domain     Nodes     Int Pt      Elems      Elems
   4          4          4          3          5

           Key nodes of subdomain nr  4
Coords           1          2          3          4
  X           3.00       5.00       5.00       3.00
  Y           3.00       3.00       4.00       4.00
 - - - - - - - - For file example - - - - - - - -
 Sub-       Elem       Elem       Rows       Cols
 Domain     Nodes     Int Pt      Elems      Elems
   5          4          4          3          2

           Key nodes of subdomain nr  5
Coords           1          2          3          4
  X           5.00       7.00       7.00       5.00
  Y           3.00       3.00       4.00       4.00
```

Boundary Condition Data - - - - - - - - - - - - -

Bound Strip	Type Cond	Value of Cond 1st Val	2nd Val	Side Nr	Element nrs Low	High	Incr
1	1	15.00	0.00	4	1	17	8
2	1	30.00	0.00	2	41	55	7

* * * * RESULTS * * * *

Nr of nodes is 74 Nr of elems is 55

node nr	x-coord	y-coord	U
1	1.0000	1.0000	15.0000
2	1.6667	1.0000	16.7397
3	2.3333	1.0000	18.4553
4	3.0000	1.0000	19.9991
5	3.4000	1.0000	20.7382
6	3.8000	1.0000	21.2974
7	4.2000	1.0000	21.6815
8	4.6000	1.0000	21.9055
9	5.0000	1.0000	21.9793
10	1.0000	1.3333	15.0000
11	1.6667	1.3333	16.7390
12	2.3333	1.3333	18.4718
13	3.0000	1.3333	20.0566
14	3.4000	1.3333	20.8046
15	3.8000	1.3333	21.3587
16	4.2000	1.3333	21.7360
17	4.6000	1.3333	21.9570
18	5.0000	1.3333	22.0302
19	1.0000	1.6667	15.0000
20	1.6667	1.6667	16.7349
21	2.3333	1.6667	18.5053
22	3.0000	1.6667	20.2479
23	3.4000	1.6667	21.0157
24	3.8000	1.6667	21.5399
25	4.2000	1.6667	21.8958
26	4.6000	1.6667	22.1104
27	5.0000	1.6667	22.1824
28	1.0000	2.0000	15.0000
29	1.6667	2.0000	16.7452
30	2.3333	2.0000	18.4559
31	3.0000	2.0000	20.6899
32	3.4000	2.0000	21.3872
33	3.8000	2.0000	21.8225
34	4.2000	2.0000	22.1528

35	4.6000	2.0000	22.3601
36	5.0000	2.0000	22.4396
37	3.0000	2.5000	22.0092
38	3.4000	2.5000	22.0714
39	3.8000	2.5000	22.3549
40	4.2000	2.5000	22.6621
41	4.6000	2.5000	22.9493
42	5.0000	2.5000	23.0102
43	3.0000	3.0000	22.5774
44	3.4000	3.0000	22.6562
45	3.8000	3.0000	22.8629
46	4.2000	3.0000	23.1927
47	4.6000	3.0000	23.6290
48	5.0000	3.0000	24.3566
49	6.0000	3.0000	27.4485
50	7.0000	3.0000	30.0000
51	3.0000	3.3333	22.8329
52	3.4000	3.3333	22.9045
53	3.8000	3.3333	23.1180
54	4.2000	3.3333	23.4722
55	4.6000	3.3333	23.9958
56	5.0000	3.3333	24.7429
57	6.0000	3.3333	27.3826
58	7.0000	3.3333	30.0000
59	3.0000	3.6667	22.9841
60	3.4000	3.6667	23.0569
61	3.8000	3.6667	23.2764
62	4.2000	3.6667	23.6508
63	4.6000	3.6667	24.1981
64	5.0000	3.6667	24.9337
65	6.0000	3.6667	27.3781
66	7.0000	3.6667	30.0000
67	3.0000	4.0000	23.0347
68	3.4000	4.0000	23.1080
69	3.8000	4.0000	23.3304
70	4.2000	4.0000	23.7108
71	4.6000	4.0000	24.2628
72	5.0000	4.0000	24.9922
73	6.0000	4.0000	27.3811
74	7.0000	4.0000	30.0000

Data File "Example.ctr"

Data File "Example.ctr"

------CONTOUR DATA FOR RUN example-----

CONTOUR VALUE	X-COORD	Y-COORD
15.0000	1.0000	1.0000
15.0000	1.0000	2.0000
15.3000	1.1150	1.0000
15.3000	1.1150	1.3333
15.3000	1.1153	1.6667
15.3000	1.1146	2.0000
15.6000	1.2299	1.0000
15.6000	1.2300	1.3333
15.6000	1.2306	1.6667
15.6000	1.2292	2.0000
15.9000	1.3449	1.0000
15.9000	1.3450	1.3333
15.9000	1.3458	1.6667
15.9000	1.3438	2.0000
16.2000	1.4598	1.0000
16.2000	1.4600	1.3333
16.2000	1.4611	1.6667
16.2000	1.4584	2.0000
16.5000	1.5748	1.0000
16.5000	1.5750	1.3333
16.5000	1.5764	1.6667
16.5000	1.5730	2.0000
16.8000	1.6901	1.0000
16.8000	1.6901	1.3333
16.8000	1.6912	1.6667
16.8000	1.6880	2.0000
17.1000	1.8067	1.0000
17.1000	1.8055	1.3333
17.1000	1.8042	1.6667
17.1000	1.8049	2.0000
17.4000	1.9232	1.0000
17.4000	1.9210	1.3333
17.4000	1.9171	1.6667
17.4000	1.9219	2.0000
17.7000	2.0398	1.0000
17.7000	2.0364	1.3333
17.7000	2.0301	1.6667
17.7000	2.0388	2.0000
18.0000	2.1564	1.0000
18.0000	2.1518	1.3333
18.0000	2.1431	1.6667
18.0000	2.1557	2.0000

18.3000	2.2730	1.0000
18.3000	2.2672	1.3333
18.3000	2.2560	1.6667
18.3000	2.2726	2.0000
18.6000	2.3958	1.0000
18.6000	2.3872	1.3333
18.6000	2.3696	1.6667
18.6000	2.3763	2.0000
18.9000	2.5254	1.0000
18.9000	2.5134	1.3333
18.9000	2.4843	1.6667
18.9000	2.4659	2.0000
19.2000	2.6549	1.0000
19.2000	2.6396	1.3333
19.2000	2.5991	1.6667
19.2000	2.5554	2.0000
19.5000	2.7845	1.0000
19.5000	2.7658	1.3333
19.5000	2.7139	1.6667
19.5000	2.6449	2.0000
19.8000	2.9140	1.0000
19.8000	2.8920	1.3333
19.8000	2.8286	1.6667
19.8000	2.7344	2.0000
20.1000	3.0546	1.0000
20.1000	3.0232	1.3333
20.1000	3.0000	1.4089
20.1000	2.9434	1.6667
20.1000	2.8240	2.0000
20.4000	3.2170	1.0000
20.4000	3.1836	1.3333
20.4000	3.0792	1.6667
20.4000	3.0000	1.7813
20.4000	2.9135	2.0000
20.7000	3.3793	1.0000
20.7000	3.3441	1.3333
20.7000	3.2355	1.6667
20.7000	3.0058	2.0000
20.7000	3.0000	2.0038
21.0000	3.5873	1.0000
21.0000	3.5411	1.3333
21.0000	3.4000	1.6418
21.0000	3.3918	1.6667
21.0000	3.1779	2.0000
21.0000	3.0000	2.1175
21.3000	3.8027	1.0000
21.3000	3.8000	1.0141
21.3000	3.7576	1.3333
21.3000	3.6169	1.6667
21.3000	3.4000	1.9217
21.3000	3.3500	2.0000

21.3000	3.0000	2.2312
21.6000	4.1151	1.0000
21.6000	4.0558	1.3333
21.6000	3.8675	1.6667
21.6000	3.8000	1.7375
21.6000	3.5955	2.0000
21.6000	3.4000	2.1555
21.6000	3.0000	2.3449
21.9000	4.5901	1.0000
21.9000	4.4968	1.3333
21.9000	4.2078	1.6667
21.9000	4.2000	1.6721
21.9000	3.8939	2.0000
21.9000	3.8000	2.0728
21.9000	3.4000	2.3747
21.9000	3.0000	2.4586
22.2000	3.0000	2.6679
22.2000	3.4000	2.6099
22.2000	3.5814	2.5000
22.2000	3.8000	2.3545
22.2000	4.2000	2.0464
22.2000	4.2911	2.0000
22.2000	4.6000	1.7863
22.2000	5.0000	1.6894
22.5000	3.0000	2.9319
22.5000	3.4000	2.8664
22.5000	3.8000	2.6428
22.5000	3.9889	2.5000
22.5000	4.2000	2.3408
22.5000	4.6000	2.1187
22.5000	5.0000	2.0529
22.8000	3.0000	3.2904
22.8000	3.4000	3.1930
22.8000	3.6782	3.0000
22.8000	3.8000	2.9381
22.8000	4.2000	2.6299
22.8000	4.3920	2.5000
22.8000	4.6000	2.3733
22.8000	5.0000	2.3158
23.1000	5.0000	2.5333
23.1000	4.6000	2.6108
23.1000	4.2000	2.9126
23.1000	4.0876	3.0000
23.1000	3.8000	3.3098
23.1000	3.7662	3.3333
23.1000	3.4786	3.6667
23.1000	3.4000	3.9478
23.1000	3.3563	4.0000
23.4000	5.0000	2.6447
23.4000	4.6000	2.8315
23.4000	4.3900	3.0000

23.4000	4.2000	3.2472
23.4000	4.1185	3.3333
23.4000	3.9320	3.6667
23.4000	3.8732	4.0000
23.7000	5.0000	2.7562
23.7000	4.6390	3.0000
23.7000	4.6000	3.0645
23.7000	4.3740	3.3333
23.7000	4.2359	3.6667
23.7000	4.2000	3.9400
23.7000	4.1887	4.0000
24.0000	5.0000	2.8676
24.0000	4.8040	3.0000
24.0000	4.6023	3.3333
24.0000	4.6000	3.3403
24.0000	4.4552	3.6667
24.0000	4.4096	4.0000
24.3000	5.0000	2.9790
24.3000	4.9689	3.0000
24.3000	4.7629	3.3333
24.3000	4.6554	3.6667
24.3000	4.6204	4.0000
24.6000	5.0787	3.0000
24.6000	5.0000	3.2100
24.6000	4.9235	3.3333
24.6000	4.8186	3.6667
24.6000	4.7849	4.0000
24.9000	5.1757	3.0000
24.9000	5.0595	3.3333
24.9000	5.0000	3.6078
24.9000	4.9817	3.6667
24.9000	4.9494	4.0000
25.2000	5.2728	3.0000
25.2000	5.1732	3.3333
25.2000	5.1090	3.6667
25.2000	5.0870	4.0000
25.5000	5.3698	3.0000
25.5000	5.2868	3.3333
25.5000	5.2317	3.6667
25.5000	5.2126	4.0000
25.8000	5.4668	3.0000
25.8000	5.4005	3.3333
25.8000	5.3544	3.6667
25.8000	5.3381	4.0000
26.1000	5.5639	3.0000
26.1000	5.5141	3.3333
26.1000	5.4771	3.6667
26.1000	5.4637	4.0000
26.4000	5.6609	3.0000
26.4000	5.6278	3.3333
26.4000	5.5999	3.6667

26.4000	5.5893	4.0000
26.7000	5.7579	3.0000
26.7000	5.7414	3.3333
26.7000	5.7226	3.6667
26.7000	5.7149	4.0000
27.0000	5.8549	3.0000
27.0000	5.8551	3.3333
27.0000	5.8453	3.6667
27.0000	5.8404	4.0000
27.3000	5.9520	3.0000
27.3000	5.9687	3.3333
27.3000	5.9681	3.6667
27.3000	5.9660	4.0000
27.6000	6.0594	3.0000
27.6000	6.0831	3.3333
27.6000	6.0846	3.6667
27.6000	6.0836	4.0000
27.9000	6.1769	3.0000
27.9000	6.1977	3.3333
27.9000	6.1991	3.6667
27.9000	6.1981	4.0000
28.2000	6.2945	3.0000
28.2000	6.3123	3.3333
28.2000	6.3135	3.6667
28.2000	6.3127	4.0000
28.5000	6.4121	3.0000
28.5000	6.4269	3.3333
28.5000	6.4279	3.6667
28.5000	6.4272	4.0000
28.8000	6.5297	3.0000
28.8000	6.5415	3.3333
28.8000	6.5423	3.6667
28.8000	6.5418	4.0000
29.1000	6.6473	3.0000
29.1000	6.6561	3.3333
29.1000	6.6567	3.6667
29.1000	6.6563	4.0000
29.4000	6.7648	3.0000
29.4000	6.7708	3.3333
29.4000	6.7711	3.6667
29.4000	6.7709	4.0000
29.7000	6.8824	3.0000
29.7000	6.8854	3.3333
29.7000	6.8856	3.6667
29.7000	6.8854	4.0000
30.0000	7.0000	3.0000
30.0000	7.0000	3.3333
30.0000	7.0000	3.6667
30.0000	7.0000	4.0000
30.0000	7.0000	3.0000
30.0000	7.0000	4.0000

Data File "Example.flo"

Data File "Example.flo"

------FLOW-LINE DATA FOR FILE example-----

CONTOUR #	X-COORD	Y-COORD
1.00	1.11	1.17
2.00	1.12	1.33
3.00	1.12	1.50
4.00	1.12	1.66
5.00	1.11	1.83
1.00	1.23	1.17
2.00	1.23	1.33
3.00	1.23	1.50
4.00	1.23	1.66
5.00	1.23	1.83
1.00	1.34	1.17
2.00	1.35	1.33
3.00	1.35	1.50
4.00	1.35	1.66
5.00	1.34	1.83
1.00	1.46	1.17
2.00	1.46	1.33
3.00	1.46	1.50
4.00	1.46	1.66
5.00	1.46	1.83
1.00	1.57	1.16
2.00	1.58	1.33
3.00	1.58	1.50
4.00	1.58	1.66
5.00	1.57	1.83
1.00	1.69	1.17
2.00	1.69	1.33
3.00	1.69	1.50
4.00	1.69	1.66
5.00	1.69	1.83
1.00	1.81	1.17
2.00	1.81	1.33
3.00	1.80	1.50
4.00	1.80	1.66
5.00	1.80	1.83
1.00	1.92	1.17
2.00	1.92	1.33
3.00	1.92	1.50
4.00	1.92	1.66
5.00	1.92	1.83
1.00	2.04	1.17
2.00	2.04	1.33
3.00	2.03	1.50

4.00	2.03	1.66
5.00	2.03	1.83
1.00	2.15	1.17
2.00	2.15	1.33
3.00	2.15	1.50
4.00	2.14	1.66
5.00	2.15	1.83
1.00	2.27	1.17
2.00	2.27	1.33
3.00	2.26	1.51
4.00	2.26	1.66
5.00	2.26	1.82
1.00	2.39	1.17
2.00	2.39	1.33
3.00	2.38	1.51
4.00	2.37	1.67
5.00	2.37	1.81
1.00	2.52	1.17
2.00	2.51	1.33
3.00	2.50	1.52
4.00	2.48	1.67
5.00	2.48	1.81
1.00	2.65	1.18
2.00	2.64	1.35
3.00	2.62	1.53
4.00	2.60	1.69
5.00	2.58	1.83
1.00	2.77	1.19
2.00	2.76	1.37
3.00	2.73	1.55
4.00	2.71	1.71
5.00	2.68	1.85
1.00	2.90	1.19
2.00	2.88	1.39
3.00	2.85	1.57
4.00	2.81	1.73
5.00	2.77	1.87
1.00	3.04	1.21
2.00	3.00	1.41
3.00	2.96	1.59
4.00	2.91	1.77
5.00	2.86	1.89
1.00	3.19	1.22
2.00	3.15	1.45
3.00	3.08	1.67
4.00	2.99	1.80
5.00	2.94	1.93
1.00	3.35	1.23
2.00	3.29	1.50
3.00	3.18	1.74
4.00	3.10	1.87

93-058

BACKGROUND INFORMATION

AUTHOR: Craig L. Arnold

RANK: 1st Lieutenant, United States Air Force

TITLE: Numerical Solution of a Second Order Boundary Problem for Two Dimensions Using Galerkin Approximations.

TOTAL PAGES: 148

TERM: Spring 1993

DEGREE AWARDED: Masters of Science in Structures and Foundations

UNIVERSITY: University of Central Florida
Orlando, FL 32816

ABSTRACT

This report outlines the development and use of the program "LAPLACE". LAPLACE is capable of solving a second order two dimensional boundary value problem, employing graphics to assist in mesh generation and solution presentation.

Galerkin approximation methods, along with the development of a finite element mesh, permit the program to calculate nodal results over the domain of the problem. The use of these nodal solutions with additional subroutines allows for the computation of equipotential lines and lines perpendicular to the equipotential lines.

The current format of this program solves Laplace's Equation. Nodal solutions to Laplace's Equation are calculated over the domain of the problem and used as the basis for the generation of equipotential lines and their perpendiculars. Equipotential lines are interpreted as contours and their perpendiculars represent flow lines for the solution to Laplace's Equation. These lines are used in combination to develop a flow net over the domain of the problem and this flow net is graphically displayed.

This program was written with the capability of solving several types of second order two dimensional boundary value problems. The calculation of solutions to other second order two dimensional boundary value problems is accomplished by entering the appropriate functional coefficients of the differential equation into one subroutine.

BIBLIOGRAPHY

Becker, E. B. et al. Finite Elements An Introduction. Volume I. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

Chapra, Steven C. and Canale, Raymond P. Numerical Methods For Engineers. New York: McGraw-Hill Book Company, 1985

Gladwell, I. and Wait, R. A Survey of Numerical Methods for Partial Differential Equations. Great Britain: Thomas Litho Ltd, 1979

Hughes, Thomas J. R. The Finite Element Method. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

Johnson, Claes. Numerical Solution of Partial Differential Equations by the Finite Element Method. New York: Cambridge University Press, 1987.

Kreyszig, Erwin. Advanced Engineering Mathematics. New York: John Wiley and Sons, 1983

Microsoft Corporation. DOS 5.0 Reference Manual. U.S.A., 1991.

Microsoft Corporation. Microsoft Basic, Basic Language Reference. U.S.A., 1989.

Segerlind, Larry J. Applied Finite Element Analysis, 2nd ed. New York: John Wiley and Sons, 1984.

Weaver, William Jr. and Paul R. Johnston. Finite Elements for Structural Analysis. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

5.00	3.00	2.00
1.00	3.55	1.26
2.00	3.44	1.56
3.00	3.30	1.81
4.00	3.21	1.95
5.00	3.05	2.08
1.00	3.75	1.34
2.00	3.60	1.69
3.00	3.41	1.91
4.00	3.28	2.05
5.00	3.10	2.16
1.00	3.98	1.47
2.00	3.74	1.81
3.00	3.53	2.05
4.00	3.35	2.18
5.00	3.16	2.27
1.00	4.20	1.67
2.00	3.89	2.00
3.00	3.64	2.19
4.00	3.46	2.33
5.00	3.19	2.42
1.00	4.38	1.94
2.00	4.03	2.18
3.00	3.77	2.38
4.00	3.57	2.51
5.00	3.22	2.64
1.00	4.52	2.16
2.00	4.17	2.36
3.00	3.90	2.57
4.00	3.68	2.71
5.00	3.26	2.89
1.00	4.60	2.38
2.00	4.31	2.55
3.00	4.04	2.75
4.00	3.83	2.91
5.00	3.45	3.16
1.00	4.64	2.60
2.00	4.44	2.73
3.00	4.18	2.93
4.00	4.01	3.09
5.00	3.71	3.39
1.00	4.72	2.78
2.00	4.55	2.87
3.00	4.34	3.06
4.00	4.19	3.26
5.00	4.00	3.55
1.00	4.80	2.89
2.00	4.64	3.00
3.00	4.50	3.18
4.00	4.37	3.34
5.00	4.24	3.66

1.00	4.85	2.97
2.00	4.76	3.07
3.00	4.64	3.27
4.00	4.56	3.43
5.00	4.45	3.69
1.00	4.95	3.03
2.00	4.88	3.15
3.00	4.76	3.33
4.00	4.72	3.48
5.00	4.65	3.72
1.00	5.05	3.08
2.00	4.99	3.22
3.00	4.91	3.38
4.00	4.86	3.53
5.00	4.81	3.73
1.00	5.14	3.11
2.00	5.09	3.25
3.00	5.04	3.41
4.00	5.01	3.56
5.00	4.97	3.75
1.00	5.23	3.14
2.00	5.19	3.29
3.00	5.15	3.43
4.00	5.12	3.59
5.00	5.10	3.76
1.00	5.33	3.17
2.00	5.29	3.31
3.00	5.27	3.45
4.00	5.24	3.61
5.00	5.23	3.77
1.00	5.43	3.19
2.00	5.40	3.33
3.00	5.38	3.47
4.00	5.36	3.63
5.00	5.35	3.77
1.00	5.53	3.21
2.00	5.51	3.35
3.00	5.50	3.49
4.00	5.48	3.64
5.00	5.47	3.78
1.00	5.64	3.22
2.00	5.63	3.36
3.00	5.61	3.50
4.00	5.60	3.65
5.00	5.60	3.79
1.00	5.75	3.23
2.00	5.74	3.37
3.00	5.73	3.51
4.00	5.72	3.66
5.00	5.72	3.79
1.00	5.86	3.23

2.00	5.85	3.37
3.00	5.85	3.51
4.00	5.85	3.67
5.00	5.84	3.79
1.00	5.96	3.23
2.00	5.97	3.37
3.00	5.97	3.52
4.00	5.97	3.66
5.00	5.97	3.79
1.00	6.08	3.22
2.00	6.08	3.37
3.00	6.08	3.51
4.00	6.08	3.66
5.00	6.08	3.79
1.00	6.19	3.21
2.00	6.20	3.37
3.00	6.20	3.51
4.00	6.20	3.66
5.00	6.20	3.79
1.00	6.31	3.21
2.00	6.31	3.37
3.00	6.31	3.51
4.00	6.31	3.66
5.00	6.31	3.79
1.00	6.42	3.20
2.00	6.43	3.37
3.00	6.43	3.51
4.00	6.43	3.66
5.00	6.43	3.79
1.00	6.54	3.19
2.00	6.54	3.37
3.00	6.54	3.51
4.00	6.54	3.66
5.00	6.54	3.79
1.00	6.65	3.19
2.00	6.66	3.37
3.00	6.66	3.52
4.00	6.66	3.66
5.00	6.66	3.79
1.00	6.77	3.19
2.00	6.77	3.37
3.00	6.77	3.51
4.00	6.77	3.66
5.00	6.77	3.80
1.00	6.88	3.19
2.00	6.89	3.37
3.00	6.89	3.51
4.00	6.89	3.66
5.00	6.89	3.79

LIST OF REFERENCES

1. Becker, E. B. et al. Finite Elements An Introduction. Volume I. Englewood Cliffs,New Jersey: Prentice-Hall, 1981.

2. Chapra, Steven C. and Canale, Raymond P. Numerical Methods For Engineers. New York: McGraw-Hill Book Company, 1985

3. Chandrupatla, Tirupathi R. and Ashok D. Belegundu. Introduction to Finite Elements in Engineering. Englwood Cliffs, N.J.: Prentice Hall, 1991.

4. Gladwell, I. and Wait, R. A Survey of Numerical Methods for Partial Differential Equations. Great Britain: Thomas Litho Ltd, 1979

5. Hinton, E. and D. R. J. Owen. Finite Element Programming. New York: Academic Press, 1977.

6. Hughes, Thomas J. R. The Finite Element Method. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

7. Johnson, Claes. Numerical Solution of Partial Differential Equations by the Finite Element Method. New York: Cambridge University Press, 1987.

8. Kreyszig, Erwin. Advanced Engineering Mathematics. New York: John Wiley and Sons, 1983

9. Microsoft Corporation. DOS 5.0 Reference Manual. U.S.A., 1991.

10. Microsoft Corporation. Microsoft Basic, Basic Language Reference. U.S.A., 1989.

11. Microsoft Corporation. Microsoft Basic, Programmer's Guide. U.S. A., 1987.

12. Potts, J. Frank and J. Walter Oler. Finite Element Applications with Microcomputers. Englewood Cliffs, N.J.: Prentice-Hall, 1989.

13. Segerlind, Larry J. Applied Finite Element Analysis, 2nd ed. New York: John Wiley and Sons, 1984.

14. Szabo, Barna and Ivo Babuska. Finite Element Analysis. New York: John Wiley and Sons, 1991.

15. Weaver, William Jr. and Paul R. Johnston. Finite Elements for Structural Analysis. Englewood Cliffs, N.J.: Prentice-Hall, 1984.